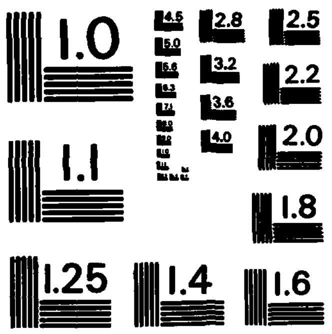
AD-A121 886 WAFER SCALE INTEGRATION OF PARALLEL PROCESSORS(U) 1/3
PURDUE UNIV LAFAYETTE IN DEPT OF COMPUTER SCIENCES
K S HEDLUND NOV 82 CSD-TR-411 N00014-80-K-0816
UNCLASSIFIED F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
10
11.2
12.5
14
16
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
1.1
1.25
1.4
1.6

12

BLUE CHiP

Wafer Scale Integration of Parallel Processors

by

Kye Sherrick Hedlund

The BLUE CHiP Project
Purdue University
Department of Computer Sciences
Math Sciences Building
West Lafayette, Indiana 47907

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CSD-TR-411	2. GOVT ACCESSION NO. AD-A121886	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Wafer Scale Integration of Parallel Processors		5. TYPE OF REPORT & PERIOD COVERED Technical, Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Kye Sherrick Hedlund		8. CONTRACT OR GRANT NUMBER(s) N00014-80-K-0816 N00014-81-K-0360
9. PERFORMING ORGANIZATION NAME AND ADDRESS Purdue University Department of Computer Sciences West Lafayette, Indiana 47907		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Task SRO-100
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Information Systems Program Arlington, Virginia 22217		12. REPORT DATE November, 1982
		13. NUMBER OF PAGES 246
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Distribution of this report is unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

wafer scale integration, VLSI, price model, CHiP computer, switch lattice, two level hierarchy, reflective switch, high parallel computers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Integrated circuit size (and hence complexity) is limited by the fact that chips created using current design techniques will not function correctly in the presence of even a single ~~circuit defect.~~ This research examines the problem of constructing chips up to the size of the wafer (wafer scale integration) that operate correctly despite the occurrence of such flaws. We concentrate on a particular family of parallel

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (Continued)

processors, configurable, highly parallel (CHiP) processors.

The key problem in the implementation of wafer scale integration is structuring the wafer so that only the functional PEs are connected together. A methodology, the two level hierarchy, that efficiently and economically solves the structuring problem for CHiP processors is presented. The principle elements are the use of column exclusion with high yield building blocks that contain redundant components. This approach limits the performance degradation due to structuring and allows the structuring problem to be solved with tractable computational effort.

Since the yield of building blocks must be high for the two level hierarchy to be a practical approach, yield phenomena are investigated in detail. ~~A model of the~~ integrated circuit manufacturing process is developed that predicts circuit yield and the probability distribution of manufacturing defects. These results are applied to the analysis of parallel processors in which several PEs occupy a single chip. In addition, they are used to design the building blocks meeting the requirements of the column exclusion strategy.

It was shown that these building blocks can be assembled into a wafer scale CHiP processor. With current technology, it is possible to fabricate a wafer scale system with 250 to 300 PEs. This represents a truly large parallel machine. Furthermore, this machine is highly robust to faults occurring during the machine's lifetime, consumes a manageable amount of power and can be efficiently tested.

Although the techniques for implementing wafer scale integration were developed for CHiP processors, they can be applied to other system composed of uniform parts.

Wafer Scale Integration of Parallel Processors

A Thesis

by

Kye Sherrick Hedlund

This work is part of the Blue CHiP Project. It is supported in part by the Office of Naval Research Contracts N00014-80-K-0816 and N00014-81-K-0360. The latter is Task SRO-100.

ACKNOWLEDGMENTS

I would like to express my strongest thanks to my advisor, Lawrence Snyder, for his tireless guidance and support throughout the course of this study. Thanks are also extended to the other members of my research committee, Peter Denning, Dennis Gannon and Gerold Neudeck, for their time and effort.

Special thanks go to my colleagues Dan Reed, David Mount and Ching Hsiao for their listening, helpful suggestions and advice, and to Julie Hanover who provided valuable assistance in preparing this manuscript.

Partial support for this work was provided by the Office of Naval Research. It is greatly appreciated.

Very special thanks are due to my mother and father for their love and encouragement.

Finally, to Becki, for her love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Hedlund, Kye Sherrick, Ph.D., Purdue University, December 1982. Wafer Scale Integration of Configurable, Highly Parallel Processors. Major Professor: Lawrence Snyder.

Integrated circuit size (and hence complexity) is limited by the fact that chips created using current design techniques will not function correctly in the presence of even a single circuit defect. This research examines the problem of constructing chips up to the size of the wafer (wafer scale integration) that operate correctly despite the occurrence of such flaws. We concentrate on a particular family of parallel processors, configurable, highly parallel (CHiP) processors.

The key problem in the implementation of wafer scale integration is structuring the wafer so that only the functional PEs are connected together. A methodology, the two level hierarchy, that efficiently and economically solves the structuring problem for CHiP processors is presented. The principle elements are the use of column exclusion with high yield building blocks that contain redundant components. This approach limits the performance degradation due to structuring and allows the structuring problem to be solved with tractable computational effort.

Since the yield of building blocks must be high for the two level hierarchy to be a practical approach, yield phenomena are investigated in detail. A model of the integrated circuit manufacturing process is developed that predicts circuit yield and the probability distribution of manufacturing defects. These results are applied to the analysis of parallel processors in which several PEs occupy a single chip. In addition, they are used to design the building blocks meeting the requirements of the column exclusion strategy.

It was shown that these building blocks can be assembled into a wafer scale CHiP processor. With current technology, it is possible to fabricate a wafer scale system with 250 to 300 PEs. This represents a truly large parallel machine. Furthermore, this machine is highly robust to faults occurring during the machine's lifetime, consumes a manageable amount of power and can be efficiently tested.

Although the techniques for implementing wafer scale integration were developed for CHiP processors, they can be applied to other system composed of uniform parts.

CHAPTER 1

INTRODUCTION

The question that motivated this research is: how can VLSI technology be utilized in the design of parallel processors? With VLSI technology it is possible to fabricate chips containing hundreds of thousands of transistors. But designing and debugging a complex integrated circuit is a lengthy and costly process. To reduce this cost and delay, it is necessary to decompose a circuit into a few different types of small substructures with simple interfaces. Technology favors replicating many copies of a simple circuit.

Consequently, this research analyzes parallel processors that are composed of a large number of simple processing elements (PEs). Each PE is a simple microprocessor and can be fabricated on a single piece of silicon. Large mainframe computers in which a single processor contains thousands of chips are not within the scope of this research.

This work concentrates on a particular family of parallel processors, configurable, highly parallel (CHiP) computers. Although the techniques for implementing wafer scale integration are developed for CHiP processors, they are entirely general and can be applied to other systems composed of uniform parts. This includes parallel processors with fixed interconnection structures, memories, etc. In Chapter 7 some extensions and generalizations of this work are discussed.

The goal of the CHiP processors considered in this work is to provide substantial parallelism at low cost. For problems that can make use of this parallelism, high performance results. We are not attempting compete with the Cray 1 nor are the machines intended to be completely general purpose. It is hoped that CHiP processors will have wide applicability, but this is an open question and a subject of further research.

1. Wafer Scale Integration

Many different architectures for parallel processors have been proposed but few large-scale parallel systems have actually been built. One reason is that a large-scale parallel processor consists of a great many components. This introduces severe practical problems of construction, wiring and reliability. If the number of individual components could be decreased, parallel processors would be far easier and cheaper to construct.

The absolute minimum number of components is reached when the entire parallel processor is fabricated on a single piece of silicon. These *wafer scale systems* have greatly reduced cost due to the increased level of integration. Reliability is higher since the connections between processors are implemented in silicon. Furthermore, there is the potential for increased preformance since data values passed between processors are not driven off the wafer.

Consider the implementation of a wafer scale system. Fabricating high density integrated circuits is a delicate process. On any given wafer, many of the chips will contain defects - errors in the circuitry such as broken wires or nonfunctional transistors. Defects are randomly distributed over the wafer surface. They are caused by imperfections inherent in the silicon or are introduced during the manufacturing process. Consequently, it is not unusual for complex circuitry to yield only 5-10% working integrated circuits from any one wafer.

To implement a wafer scale system, all chips on a wafer are tested, and then the good chips are connected together. The wafer is structured so that the presence of faulty chips is masked and only functional chips are used. This ***structuring problem*** is the key problem in the implementation of wafer scale integration (WSI). With low yield, the good chips are sparsely and irregularly distributed over the wafer surface so the key problem is to provide a highly flexible means of connecting chips.

Consider the problem of connecting functional chips in a mesh pattern. This is fundamental for constructing CHiP computers. The structuring problem is made difficult by low chip yield. For any particular good chip, it is very unlikely that all its four neighbors will also be functional; the positioning of good chips on the wafer differs from the required connection pattern - the mesh. Hence, considerable wiring may be required to connect a chip to its neighbor in the mesh.

Now suppose that most chips are functional. The good chips are distributed in a more regular pattern - one closely resembling a mesh. This simplifies the structuring problem. For example, Figure 1.1.1 shows a wafer containing a 4×5 grid of chips with only one faulty chip. A 4×4 mesh is obtained by excluding all chips in the column containing the fault. This strategy is called ***column exclusion***. The only requirement is that we can

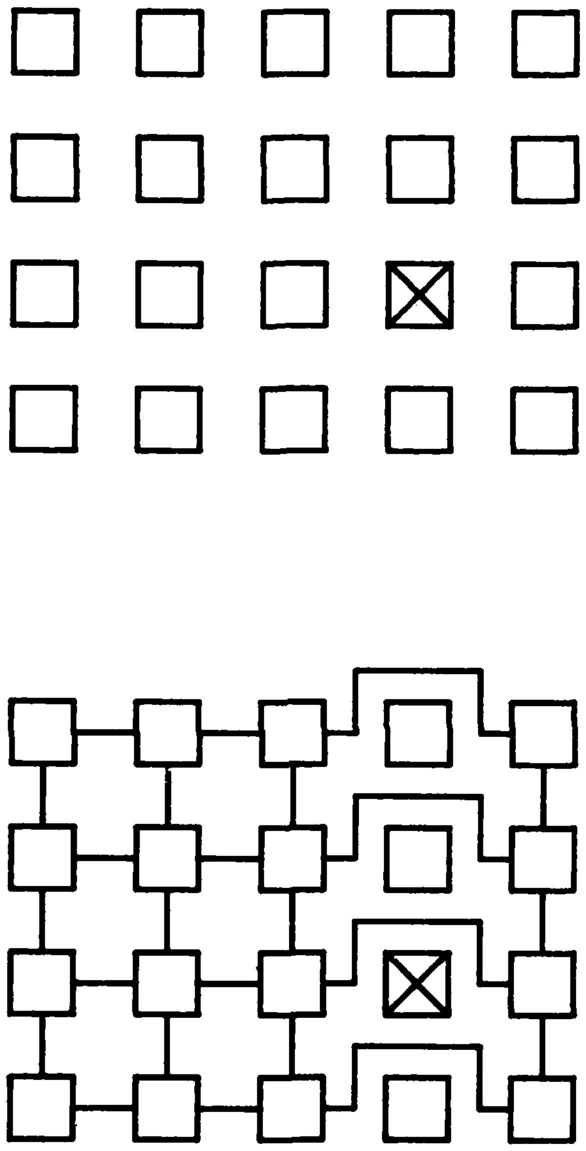


Figure 1.1.1 - Structuring by Column Exclusion

wire around faulty or unused chips. This strategy has been used in 64K memories [Cenk79, Eato81, Kokk81] and in a computer architecture on Massively Parallel Processor [Batc79].

For this simple approach to be practical, the wafer must contain very few faulty chips. But due to the nature of the integrated circuit manufacturing process, high yield is achievable only with very simple chips - much less complex than a processing element that is needed for a parallel processor.

But suppose the units patterned on the wafer are not individual processors but ***building blocks*** of a mesh. With each block contributing a small mesh of fixed size, the blocks can be assembled to form a larger mesh. For example, with a 4 × 4 grid of blocks each containing a 2 PE by 2 PE mesh, a mesh with 8 PEs on a side is formed. The key idea is that each block will contain sufficiently many redundant PEs to insure that a small, functional mesh will exist within almost every block. Virtually every block on the wafer will contribute a small subpart to the overall structure, so the structuring problem can be solved by eliminating the columns (or rows) containing the relatively rare blocks which are completely dysfunctional. This technique is practical if the blocks meet two requirements:

1) Blocks must have high yield; most blocks must contain a smaller, fully functional mesh.

2) Blocks that are unused or faulty can be "wired around" to connect the two blocks in the adjacent columns.

In the remainder of this chapter, we survey previous work on wafer scale integration and give a concise summary of the ideas behind CHiP processors. The approach to wafer scale integration using column exclusion and building blocks is discussed in more detail in Chapter 3. Since the yield of building blocks must be high, yield phenomena are investigated in Chapter 2. In Chapter 4, the yield results are used to design the building blocks of a wafer scale CHiP processor. The assembly of the blocks into a complete wafer scale system is the topic of Chapter 5. The testing of CHiP processors is discussed in Chapter 6, and the final chapter provides a brief summary of the results along with possible extensions and generalizations of this research. Figure 1.1.2 shows the interrelationships of the main concepts in this thesis. The numbers in parenthesis indicate the chapters in which the topic is discussed.

2. Previous Work on Wafer Scale Integration

Research into wafer scale integration has been conducted for over fifteen years starting with discretionary wiring. In this approach, modules (PEs, memory units, etc.) are patterned on the wafer and are individually tested by wafer probing. A wiring pattern to connect together the good modules is automatically generated. This wiring is implemented by extra levels of metal interconnections that are placed overtop the modules. The structuring problem is solved by these extra layers of customized wiring.

Discretionary wiring was strongly backed by both Texas Instruments and the Air Force. Despite strong funding and years of research, it never became a practical means of implementing WSI. There are two major problems with this approach

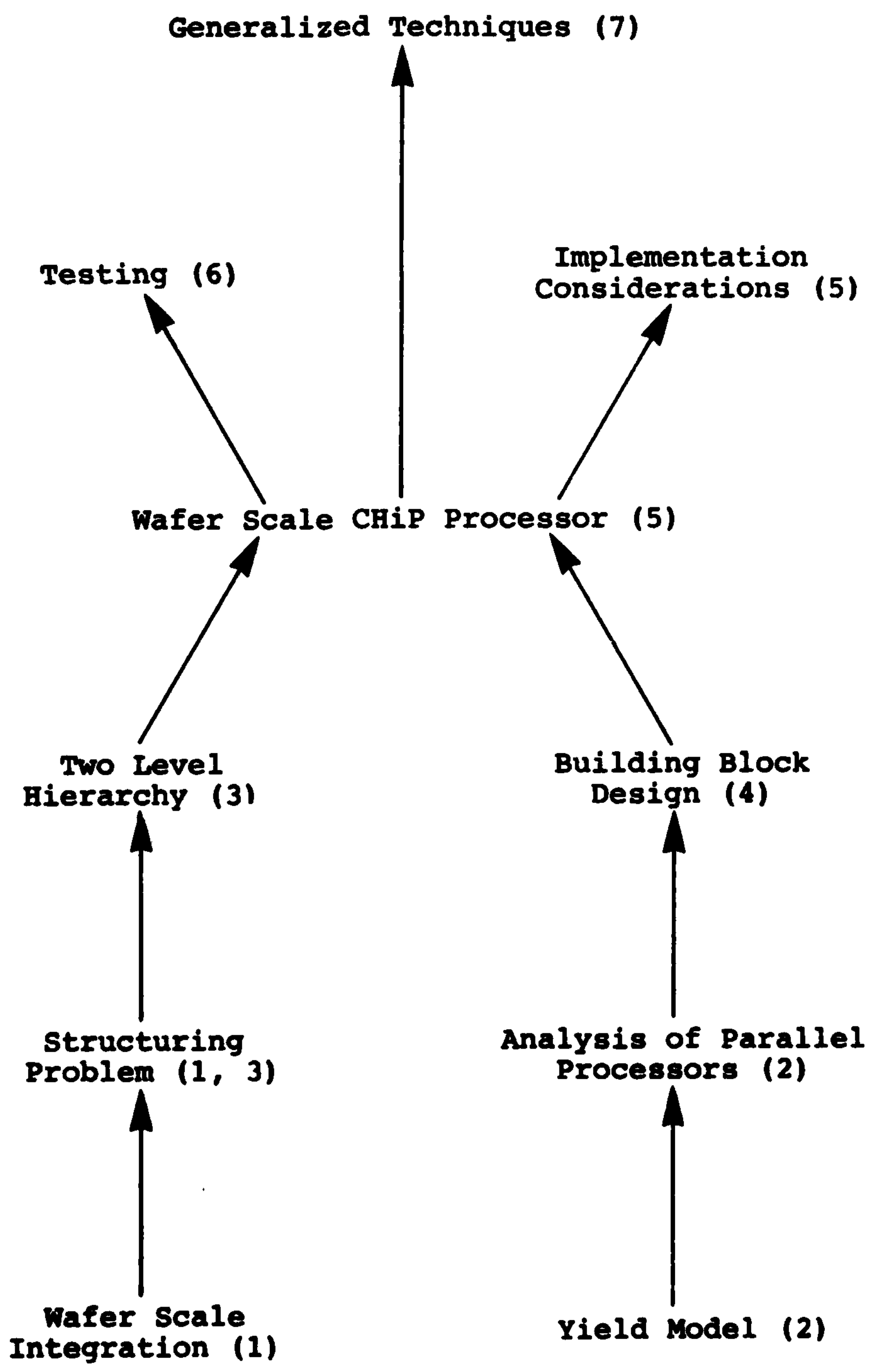

Generalized Techniques (7)
Testing (6)
Implementation Considerations (5)
Wafer Scale CHiP Processor (5)
Two Level Hierarchy (3)
Building Block Design (4)
Structuring Problem (1, 3)
Analysis of Parallel Processors (2)
Wafer Scale Integration (1)
Yield Model (2)

Figure 1.1.2 - Interrelationship of Main Concepts

- Excessive cost. Defects are randomly distributed over the wafer surface. With a large number of modules per wafer, there are an enormous number of different patterns of good and bad modules. This requires that a unique set of photolithography masks be made to define the wiring pattern for each individual wafer. This is prohibitively expensive [Aubu78].

- Faults occur in the upper levels of metalization used for structuring. The topmost levels of interconnection, as with the lower levels, are subject to faults such as poor contacts between levels and shorts to underlying levels [Aubu78, IEEE82]. These faults effect not just a single module but the entire wafer.

As these problems surfaced, researchers attempted to reduce the complexity of the custom wiring. Each level of interconnection requires two photolithography masks. One defines the wiring pattern, and the other determines the connections between levels. The initial work on discretionary wiring required two customized metalization levels and hence four unique masks for each wafer.

The pad relocation technique [Calh72] reduces the number of unique masks to one. A single, standard wiring pattern on the topmost metal level interconnects fixed position "pads" on the first level of metalization. This lower metalization level is customized for each wafer to relocate the wiring of modules to the pads. Only good modules are connected to a pad. The upper level makes a standard sequence of connections between fixed locations, the pads, and the connections between pads and modules varies in response to the defect pattern of the particular wafer. Only the mask defining the lower metalization level need be modified from wafer to wafer.

Despite this cost reduction, pad relocation did not produce reliable and economical wafer scale systems. The problems are the assumptions that the customized processing steps would be fault free and that no modules tested as good would fail during the remaining processing. It was recognized that the additional processing steps required to define the customized wiring are the Achilles heel of these approaches.

The work of Manning [Mann75] and the independent but closely related research of Aubusson [Aubu73, Aubu78] proposed solutions to the structuring problem that required no extra wafer processing steps. The essential feature of the approach is that each module can be externally programmed to connect to any of its immediate neighbors. There is an implicit switching mechanism within each module. By selectively connecting modules only to functional neighbors, a linear array of good modules can be "snaked" through the grid of modules on the wafer. Heuristics for maximizing the length of the chain were developed [Aubu78, Fuss82, Mann75].

Since no extra processing steps are required, this solves the problems that plagued discretionary wiring and pad relocation, but at the cost of flexibility. The wafer is structured only into a linear array; the solution to the structuring problem is only one dimensional.

The structuring of the wafer into a richer set of two dimensional configurations is a major problem in the implementation of wafer scale systems. Fussell and Varman [Fuss82] have presented algorithms for a priority queue and a triangular array capable of performing the multiplication of a band matrix and a vector. Koren [Kore81] developed algorithms for a binary tree and a mesh.

Recent advances in integrated circuit manufacturing may provide new methods for implementing wafer scale integration. The most promising of these is laser programming [Kuhn75, Logu80, Mano80, Wu82]. Submicron thick layers of quartz sandwich the uppermost level of metal with a lower level of metal underneath. A series of short laser pulses burns through the quartz layers to weld the two metal levels. This forms a low impedance contact.

The use of laser programming to implement wafer scale systems is under investigation at Lincoln Laboratories [Chap]. Modules are patterned on the wafer with fixed wiring corridors between them. Vertical wires are run in the first metal layer and horizontal wires in the second. Initially, the modules are unconnected. After testing, laser programming makes the connections required to interconnect the functional modules.

This technique resembles discretionary wiring. Although the wiring pattern is fixed, the connections between wires are completed after testing. But with advances in semiconductor processing technology, wiring channels can be manufactured with high reliability. Also, the laser welds form low impedance contacts with very high probability. Thus there are very few faults in the custom wiring.

However, this approach has one serious drawback. The connections made with laser programming are static; once they made they can not be changed. A wafer scale system can contain hundreds of thousands of gates and millions of transistors. During the lifetime of a system, faults are very likely to occur. It is certainly undesirable to discard an entire wafer due to a single faulty transistor. With laser programming, there is no method of reconfiguring the wafer after manufacturing. A single fault during the system lifetime may disable the entire wafer scale system.

3. Introduction to CHiP Processors

A brief introduction to CHiP processors is presented here. More detailed information can be found in [Snyd82a]. The CHiP processor is a family of architectures each constructed from three components: a collection of microprocessors, a switch lattice and a controller. The switch lattice is the most important component and the main source of differences between family members. It is composed of programmable switches connected by datapaths. The microprocessors function as the processing elements of the system. They are not directly connected to each other, but rather are inserted at regular intervals into the switch lattice. Figure 1.3.1 shows three different switch lattices. The perimeter switches are connected to external storage devices.

Each switch has local memory capable of storing several configuration settings. A configuration setting enables the switch to establish a direct, static connection between two or more of its incident datapaths. (This is circuit switching rather than packet switching.) Figure 1.3.2 shows a mesh configured CHiP processor. Switches in alternating columns are assigned the North-South configuration setting and every other row has switches set to connect East to West. The controller is responsible for loading the switch memory and the programs into the PEs. It is the supervisor of the CHiP

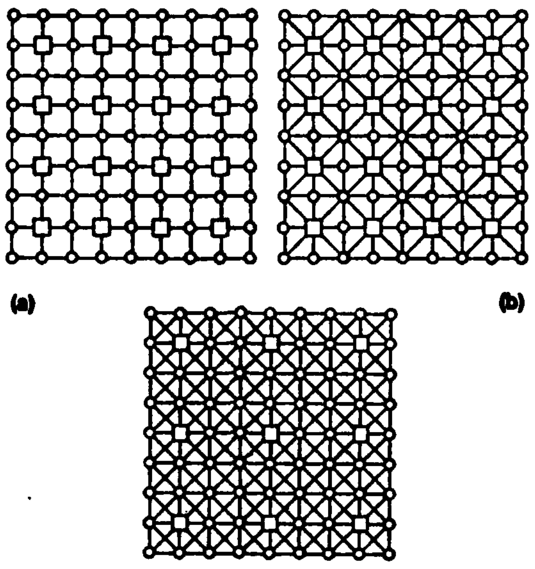


Figure 1.3.1 - Three CHiP Processors (Circles Represent Switches; Squares Represent PEs)

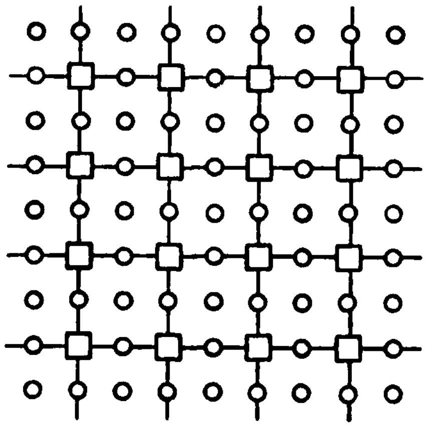


Figure 1.3.2 - Mesh Configured CHiP Processor

processor and is responsible for starting and stopping the PEs.

Members of the CHiP family are distinguished by their lattice parameters:

- degree - number of incident datapaths
- crossover - number of distinct datapath groups that a switch can simultaneously connect
- corridor width - number of switches that separate two adjacent PEs.

The lattice of Figure 1.3.1a, the ***white lattice***, is a simple CHiP structure having degree four, one crossover and a corridor width of one.

CHAPTER 2

YIELD MODEL

The implementation choices that must be made when designing a fault tolerant CHiP machine are strongly influenced by the percentage of faulty processing elements within the parallel processor. For example, greater flexibility in interconnecting the PEs may be required if a large fraction of PEs are faulty than if only a small number fail. Furthermore, redundancy can be used to increase the yield of a CHiP lattice. The amount of redundancy required to achieve a given yield depends on the mean number of faulty PEs. Consequently, a necessary prerequisite to the analysis of fault tolerant parallel processor design is to determine the number of faulty processing elements. This problem is the focus of this chapter.

This research analyzes implementations of CHiP machines in silicon. A number of PEs will be fabricated on a single area of silicon called a ***building block***. A complete CHiP machine consists of one or more building blocks. The individual building blocks may reside on separately packaged chips or, in wafer scale systems, on different portions of a single piece of silicon. Since the occurrence of defects on a silicon wafer is a random process, the exact number of faulty PEs cannot be predicted. Instead, a probability density function describes the fault process. This is the probability that a given number of defects will occur. It is dependent on many interrelated factors of design and semiconductor processing technology. A *yield model* is a mathematical model of the integrated circuit manufacturing process that relates the probability of the occurrence of defects to factors such as defect density, design rules, etc. The design parameter most directly controlled by the computer architect is the area occupied by a building block. Consequently, a yield model and the corresponding density function dependent on the silicon area will be derived below.

The starting point for the development of the yield model is a widely accepted model due to Price [Pric70]. It will be simplified to exclude factors that pertain to the fabrication process but are not under the control of the silicon architect, and some parameters will be assigned values appropriate for the implementation of CHiP machines. The end result of the modeling of the semiconductor fabrication process will be a function, $Pr(Z=m; A)$, computing the probability of exactly m defects occurring within an area of silicon, A. This function will be used to compute the expected number of defective PEs in a building block. It will be a workhorse in the analysis of the effect of fault tolerance on parallel processor design.

1. The Price Model

The starting point of our development of a yield model is the multistep Price model [Pric70] which is one of the more realistic models of integrated circuit manufacturing [Glas79, Stap76]. It has shown close agreement with empirical evidence [Glas79, Cenk81]. Underlying this model are several assumptions:

1. All point defects belong to one of k distinguishable classes of indistinguishable defects. Defects in different classes can be told apart by inspection, but within a single class, defects are indistinguishable. Each class represents the defects introduced by one critical masking step in the fabrication process. (Throughout this paper we use the terms *processing* or *fabrication step* to refer to a critical masking step, not operations such as etching, oxide growth, etc.)

2. Each of the fabrication steps is independent of the others; the number of defects introduced by the i^{th} step does not depend on the number of defects introduced by previous steps. This a direct result of the design rules. Design rules incorporate sufficient spacing between levels such as polysilicon and diffusion to insure that a minor mask misalignment will not create unwanted transistors. Furthermore, design restrictions such as not allowing contact cuts overtop gates insure that the processing at upper levels will not damage fragile portions of underlying layers. The primary consequence of this assumption is that the total number of defects is the sum of the defects introduced by each processing step.

3. The density of fatal defects is the same for each fabrication step. On the average, each processing step contributes equally to the probability of a fatal defect occurring. Yield is maximized when all steps contribute equally to the introduction of defects. Consequently, the design rules are set to insure this. For example, the metalization layer runs over rougher terrain than does the polysilicon layer. This makes metal lines more susceptible to breaks and shorts so metal line widths

and spacings are typically larger than for the polysilicon layer.

From these assumptions, we can derive the following relationship [Glas79]

$$Y = \frac{C}{\left[1 + d\, Q(r/r_0)\right]^k} \tag{1.1}$$

where Y is the yield (i.e., fraction of chips which are functional). The parameters have the following interpretations:

C	fraction of wafer area not wasted due to clustering defects
$Q(r/r_0)$	represents the effect of the design rules employed on the specific circuit. It depends on the minimum spacing, r, and an empirical threshold spacing r_0. When r approaches r_0, $Q(r/r_0) >> 1$ and the yield drops appreciably. With relaxed design rules, $r > r_0$ and $Q(r/r_0)$ approaches a limit q' with $0 < q' \leq 1$, and yield increases.
k	number of critical masking steps (*i.e.* number of defect classes)
d	defect density/chip for a single fabrication step

The above model will be modified to make it applicable specifically to the analysis of fault tolerant parallel processors. Parameters representing details of the fabrication process or the design rules will be eliminated, and specific values for other parameters will be introduced. The result will be a simplified model relating the yield to the chip area.

2. Yield Model for Analysis of Fault Tolerance

The following simplifications in the above model are made to tailor it to the analysis of fault tolerant design:

1. Only random defects are considered. (Throughout this paper, the term defect will refer to a *fatal* defect; one that causes the circuit in which it occurs to function incorrectly.) It is assumed that defects have no tendency to cluster on any portion of the wafer [Stap75, Stap76, Stap80, Stap82, Sait82]. Non-random defects are due to scratches in a photolithography mask, surface imperfections resulting from polishing, etc. Currently, the number of non-random defects per wafer can be made low (e.g., 1-2 for a 2" wafer). Improvements in processing technology and increased care in handling wafers during fabrication can reduce the number of non-random defects. Experience at Lincoln Laboratories shows that they can be virtually totally eliminated [Chap] by more careful wafer screening, increased care in wafer handling and more frequent mask inspection. Consequently, we assume $C = 1$.

2. A 4-layer process is assumed. Currently, a 3-layer process defining three levels of interconnection (diffusion, poly and metal) is common. For implementation of CHiP processors, it is highly desirable to have an additional level to facilitate the interconnection of PEs and the routing of common control and power signals (the skeleton). Since metal has the lowest RC constant, it is desirable to use an additional level of metal for the relatively long wires of the skeleton and for the wires between PEs. A two level metal process is in use by several manufactures. Thus it is reasonable to assume such a process for CHiP implementation. Consequently, we assume there are four interconnection levels (diffusion, poly and two metal layers), and we let $k = 4$.

These simplifications reduce equation 1.1 to

$$Y = \frac{1}{(1 + d\,Q(r/r_0))^4} \tag{2.1}$$

Yields vary greatly depending on the particular fabrication line, the process being run, etc. It is undesirable to have the results of this work apply only to a specific circuit or fabrication process. The results should be independent of the semiconductor processing details. Consequently, the many processing and design factors must be lumped together into a single factor. To accomplish this, rather than measure area by absolute quantities (e.g., square mils), we will introduce the concept of normalized unit area.

Yield depends on both the details of the circuit layout and the design rules employed since different layouts will have different sensitivities to variances in the design rules. In Chapter 4, the design of a "standard" PE for CHiP processors is outlined. It has an 8-bit ALU with 64 bytes of memory and a simple arithmetic oriented instruction set. This is sufficient to execute a wide variety of systolic algorithms [Snyd82a]. This is the yardstick by which PE complexity will be measured.

From one fabrication line to another, the design rule spacings of the circuit layout of the standard PE can be modified to change the yield. Relaxed design rules will increase both the yield and the area occupied by the PE while tight design rules can be used on fabrication lines with more precise manufacturing tolerances to pack more PEs into a given area. Thus the design rules and the yield can be traded off against each other (within certain limits). Depending on the particular fabrication line, the design rules are adjusted so that the standard PE is produced with predetermined yield.

- A ***normalized unit area*** (NUA) is the silicon area occupied by a 2 × 2 white lattice of standard PEs with the design rules set to achieve a 20% yield of the lattices.

(The yield for the unit area definition assumes no fault tolerance; one defect renders the chip dysfunctional). The 20% yield Figure is somewhat arbitrary but was chosen so that a normalized unit area represents a medium to medium large chip. All area measurements in this work will be in terms of normalized unit area with the understanding that the exact size of a NUA will vary from one fabrication line to another, with improvements in semiconductor technology, from nMOS to CMOS implementation, etc.

To convert equation 2.1 to units of normalized unit area, we define

s_0 = average number of defects ***per normalized unit area*** for a single processing step

We can then replace d $Q(r/r_0)$ in the yield model by A s_0

$$Y = \frac{1}{(1 + A\,s_0)^4} \tag{2.2}$$

where A is the chip area measured in NUA. The concept of unit area has eliminated the dependence on the design rules and the particular circuit being manufactured. The area of a building block will be measured ***relative*** to the area of the standard 2 × 2 white lattice.

To determine the value of s_0 , solve equation 2.2 for s_0. By definition $Y = 0.20$ at $A = 1.0$ so

$$s_0 = (0.20)^{-1/4} - 1 = 0.495 \quad \text{(defects per unit area per step)}$$

Figure 2.2.1 shows the yield as a function of the chip area measured in NUA. Note that the yield drops steeply at first then levels off at low yield. This is consistent with empirical evidence. Defects limit chip area; chips that are too large have prohibitively low yield.

Because the processing steps are assumed independent and the total number of defects is the sum of the defects introduced by each processing step, d_0, the average number of defects per normalized unit area after all four fabrication steps is

$$d_0 = 4s_0 = 1.98 \quad \text{(defects per unit area)}$$

d_0 is a fundamental quantity in the analysis of fault tolerance. From it we know the mean number of defects in a CHiP lattice of a given area - since defects are randomly distributed, the expected number of defects in area A is Ad_0 (Table 2.2.1).

3. Probability Density Function

The yield is the probability of no defects. Since we are concerned with the design of fault tolerant machines, a certain number of defects (the exact number depends on the design details) can be present without rendering the machine dysfunctional. Therefore, rather than yield, we are interested in the number of defects and their probability distribution. It is at this point that this research diverges from previous work on yield models. The design of fault tolerant CHiP processors requires a more detailed examination of the fault distribution.

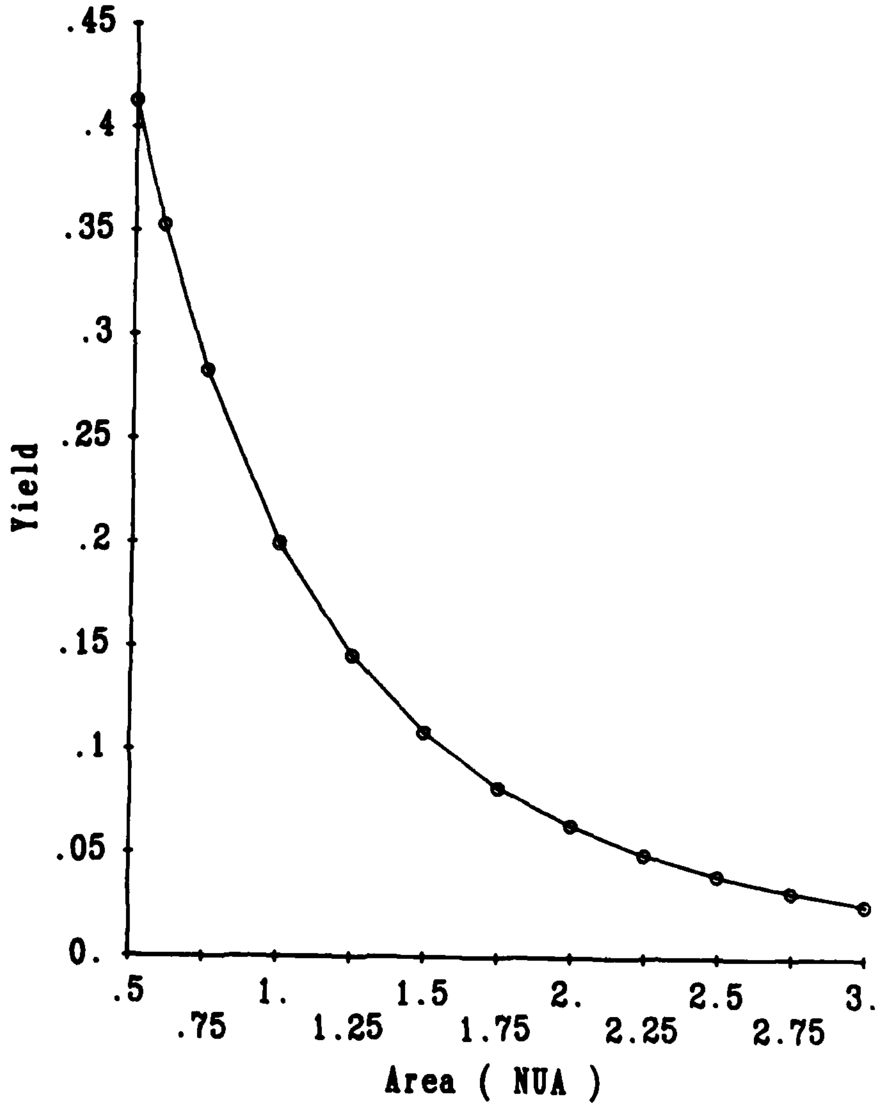


Figure 2.2.1 - Yield vs. Area (in NUA - Normalized Unit Area)

Table 2.2.1 - Expected Number of Defects as a Function of Area (in NUA)

Area (in NUA)	Expected Number of Defects
.6	1.19
.8	1.58
1.00	1.98
1.25	2.48
1.50	2.97
1.75	3.47
2.00	3.96
2.25	4.46
2.50	4.95
3.00	5.94

The probability that exactly m defects occur in a lattice of area A is denoted by $\Pr(Z=m;A)$, where Z is a random variable representing the number of defects. For a design that can accommodate up to m' defects and occupies area A, the probability that the machine is functional is $\Pr(Z \leq m';A)$. When the area is a fixed quantity, the area parameter will sometimes be omitted and the density function abbreviated as $\Pr(Z=m)$.

Let z_i be the random variable denoting the number of defects introduced by the ith processing step and Z be the number of defects after all processing steps. $\Pr(z_i=m)$ follows a geometric distribution [Glas79]

$$\Pr(z_i=m;A) = p(1-p)^m \qquad \text{with } p = \frac{1}{1 + As_0}$$

where s_0 is the defect density.

In a multistep process, total number of defects is the sum of the defects introduced by the individual processing steps. Hence, for a given area, A, $\Pr(Z=m)$ is the sum of independent and identically distributed geometric random variables. For a four step fabrication process,

$$\Pr(Z=m) = \Pr(z_1 + z_2 + z_3 + z_4 = m)$$

Summing the four independent variables, we have

$$\Pr(Z=m) = \sum_{i=0}^{m} \Pr(z_1=i) \sum_{j=0}^{m-i} \Pr(z_2=j) \sum_{k=0}^{m-i-j} \Pr(z_3=k) \Pr(z_4=m-i-j-k)$$

$$= \frac{1}{6}(m+1)(m+2)(m+3)\, p^4(1-p)^m$$

where i,j and k are the number of defects introduced by the 1st,2nd and 3rd processing steps. The derivation of this equation is given below.

Derivation - Summation of Geometric Random Variables

Assume the random variables z_1, z_2, z_3, z_4 are independent and have identical geometric distributions, $\Pr(z_i=m) = pq^m$ with $q = 1-p$. We will derive the distribution for the sum of 2, 3 and 4 of the random variables. The four variable case represents the probability of m defect as predicted by the 4 step Price yield model, the primary model used in this research.

Two Random Variables:

The m successes must be divided between the two random variables. z_1 can account for between none and all of them with z_2 making up the remainder.

$$\Pr(z_1 + z_2 = m) = \sum_{i=0}^{m} \Pr(z_1 = i) \Pr(z_2 = m-i) =$$

$$\sum_{i=0}^{m} pq^i \; pq^{m-i} = \sum_{i=0}^{m} p^2q^m = (m+1)\, p^2q^m$$

Three Random Variables:

Divide the successes into two groups, those of z_1 and those of z_2 and z_3 combined. The total number of successes, m, can be arbitrarily divided between the two groups, and the two random variable result from above can be used to evaluate $\Pr(z_2 + z_3 = m-i)$.

$$\Pr(z_1 + z_2 + z_3 = m) = \sum_{i=0}^{m} \Pr(z_1 = i) \Pr(z_2 + z_3 = m-i) =$$

$$\sum_{i=0}^{m} pq^{i}\ (m-i+1)p^{2}q^{m-i} = p^{3}q^{m}\sum_{i=0}^{m}(m-i+1) =$$

$$p^{3}q^{m}\left[\sum_{i=0}^{m}(m+1) + \sum_{i=1}^{m} i\right] = p^{3}q^{m}[(m+1)^{2} - \frac{1}{2}\ m(m+1)] =$$

$$\frac{1}{2}(m+1)\ (m+2)\ p^{3}q^{m}$$

Four Random Variables:

Analogously to the three random variable case, we partition the random variables into two groups: $\{z_1\}$ and $\{z_2, z_3, z_4\}$. The three variable result from above is employed.

$$\Pr(z_1 + z_2 + z_3 + z_4 = m) = \sum_{i=0}^{m} \Pr(z_1 = i)\ \Pr(z_1 + z_2 + z_3 = m-i) =$$

$$\sum_{i=0}^{m} pq^{i}\ \frac{1}{2}(m-i+1)\ (m-1+2)\ p^{3}q^{m-i} =$$

$$\frac{1}{2}p^{4}q^{m}\sum_{i=0}^{m}(m-i+1)\ (m-i+2) =$$

$$\frac{1}{2}p^{4}q^{m}\left[\sum_{i=0}^{m}(m^{2} + 3m + z) + \sum_{i=1}^{m}(i^{2} - (2m+3)\ i)\right] =$$

$$\frac{1}{2}p^{4}q^{m}\left[(m+1)\ (m^{2}+3m+2) + \frac{1}{6}m(m+1)\ (2m+1) - (2m+3)\ \frac{1}{2}m(m+1)\right] =$$

$$\frac{1}{6}(m+1)\ (m+2)\ (m+3)\ p^{4}q^{m}$$

■

Figure 2.3.1 and Table 2.3.1 show the probability of m defects, Pr(Z=m; A), for several different areas measured in units of NUA. It is important to observe that for smaller areas the curves peak at a very small value (*e.g.* 1 - 2) of m. This means the chances of a large number of defects is quite small.

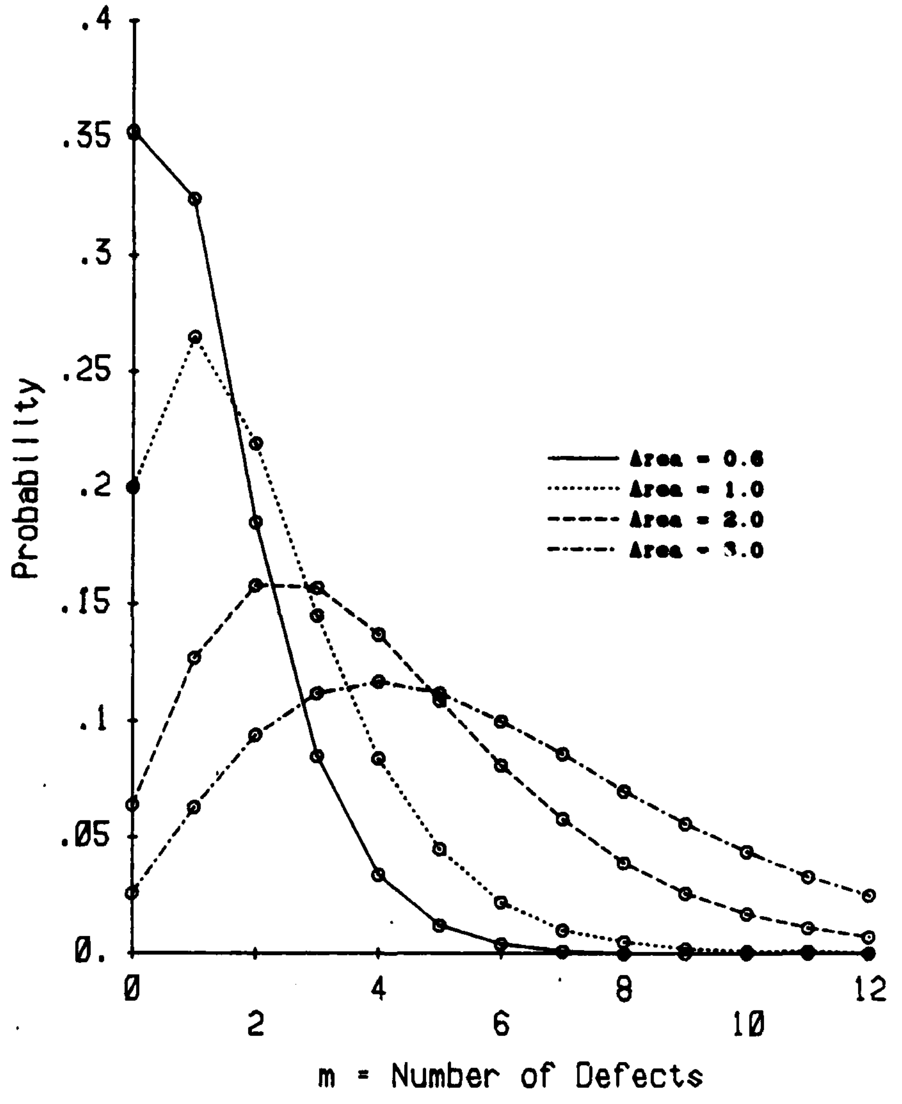


Figure 2.3.1 - Probability of m Fatal Defects as a Function of Area (in NUA)

Table 2.3.1 - Probability of m Fatal Defects as a Function of Area (in NUA)

	Pr(Z = m; A)			
number of defects(m)	Area (in NUA) 0.6	1.0	2.0	3.0
0	.353	.200	.064	.026
1	.324	.265	.127	.063
2	.185	.219	.158	.094
3	.085	.145	.157	.112
4	.034	.084	.137	.117
5	.012	.045	.109	.112
6	.004	.022	.081	.100
7	.001	.010	.058	.086
8	.000	.005	.039	.070
9	.000	.002	.026	.056
10	.000	.001	.017	.044
11	.000	.001	.011	.033
12	.000	.000	.007	.025

For example, in a unit area, the probability of 6 defects is 2% whereas a single defect occurs 27% of the time. Consequently, the cumulative probability, Pr($Z \leq m$; A), rises quickly (see Figure 2.3.2 and Table 2.3.2). This means that at low yield, even though there is a large probability of at least one defect, the number of defects is likely to be small. The yield of the whole fabrication process is the product of the yields of the individual steps. With four processing steps and under the assumption of identical yield at each step, overall yield equals the yield of an individual step to the forth power (equation 2.2). The yield of a single step is inversely proportional to the chip area. Consequently, yield decreases quickly as chip area increases (Figure 2.2.1); yield is the *product* of four identical terms. On the other hand, the probability distribution of the number of defects per chip, Z, is the *sum* of four identically distributed random variables. This exhibits a peaked distribution in which the probability of a large number of defects is small.

4. Comparison of Yield Models

In the previous sections, a multistep Price yield model was developed. Is this particular model the most appropriate? There are other yield models such as the Poisson and Gaussian models which are based on slightly different and less realistic assumptions about the semiconductor manufacturing process. However, their mathematical formulation is considerably simpler than the Price model. Are they sufficiently accurate for the types of problems we will consider? Can a good approximation be obtained with simpler mathematics? This section examines the different

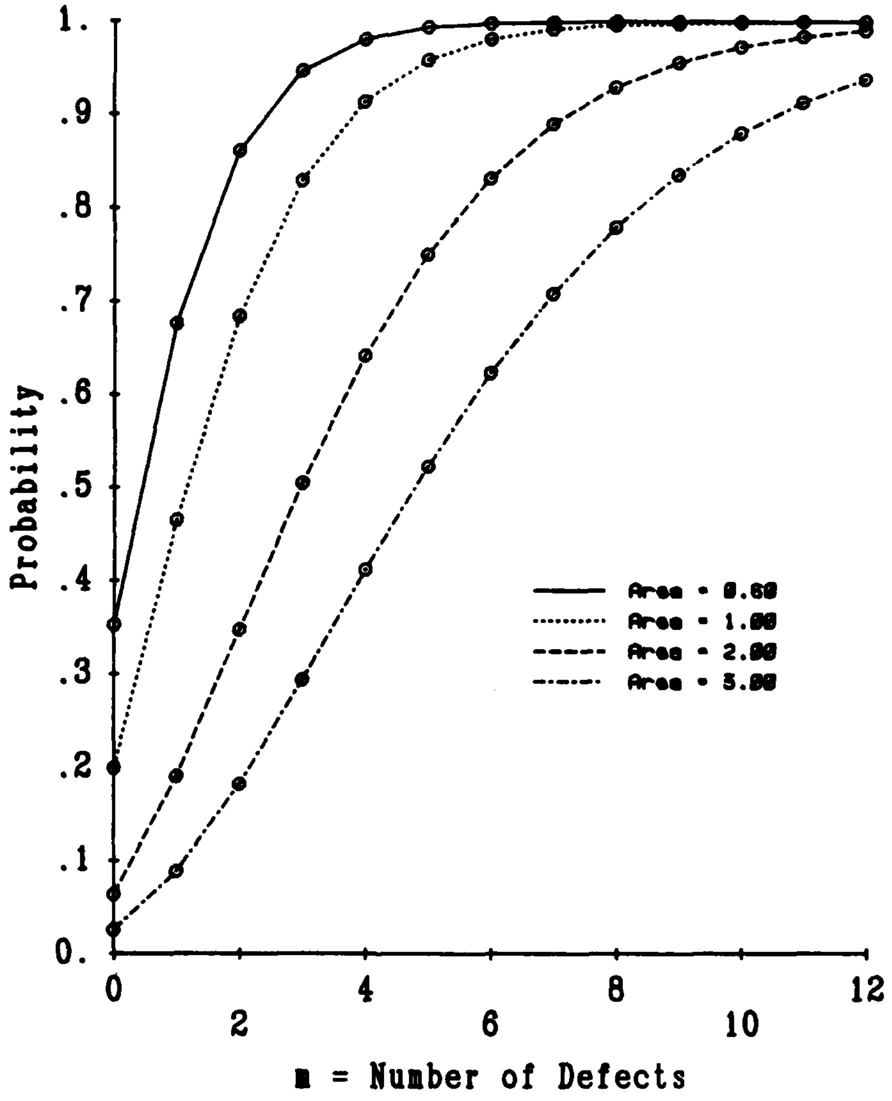


Figure 2.3.2 - Cumulative Probability of m Fatal Defects

Table 2.3.2 - Cumlative Probability of m Defects as a Function of Area (in NUA)

	Pr(Z ≤ m; A)			
number of defects(m)	Area (in NUA) 0.6	1.0	2.0	3.0
0	.353	.200	.064	.026
1	.677	.465	.191	.089
2	.862	.685	.348	.183
3	.947	.830	.505	.294
4	.981	.914	.642	.412
5	.994	.959	.751	.523
6	.998	.981	.832	.624
7	.999	.992	.890	.709
8	1.000	.996	.929	.780
9	1.000	.998	.956	.836
10	1.000	.999	.973	.880
11	1.000	1.000	.984	.913
12	1.000	1.000	.991	.938

models and compares their accuracy. The basic question is whether the increased accuracy of the Price model is worth its added complexity. It is answered affirmatively.

Figure 2.4.1 shows the relationship of the different yield models. The key underlying assumption is the distinguishability of defects. If the wafer were examined by an inspector, could each of the individual defects be told apart? The Poisson and Gaussian models assume distinguishable defects whereas the Price model assumes the defects have identical appearances. This assumption determines the form of the probability density function for the occurrence of defects. For example, consider the total number of ways m defects can occur in a set of n different chips. For many of the probabilities that will arise in applications of the yield model, this is the size of the sample space. If the defects are distinguishable, there are n^m different assignments of defects to chips whereas indistinguishable defects give only

$$\binom{m+n-1}{m} < n^m$$

placements. The different sizes of the sample space give rise to different probability distributions. Additionally, equations involving terms such as n^m generally are simpler than those involving the more complex combinatorial formulae. Consequently, the Price models are more complex and difficult to work with than the Poisson and Gaussian models.

Although they are more complex, the Price models are more realistic. They agree more closely with empirical evidence [Glas79]. Furthermore, it is unrealistic to assume that defects of similar physical cause (e.g. two oxide

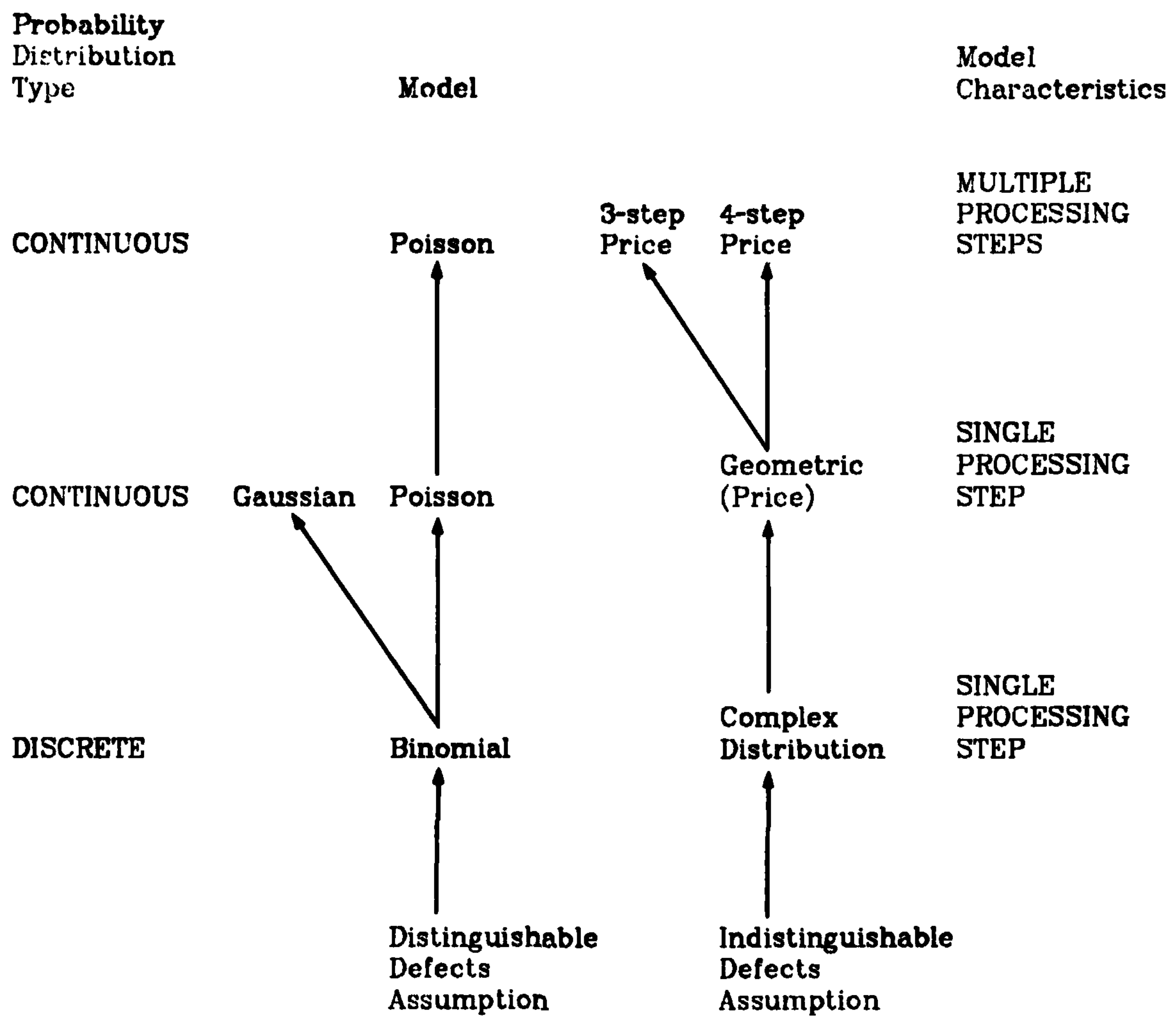


Figure 2.4.1 - Taxonomy of Yield Models

pinholes) can be told apart. However, an inspector could tell a metal short from an oxide pinhole. This supports the distinguishable classes of indistinguishable defects which underlies the Price model.

a) Distinguishable Defects

Assume each defect is unique and can be differentiated from all other defects. With M distinguishable defects distributed over N chips, the probability that any given chip contains exactly k defects after a single processing step is

$$\Pr(z=k) = \binom{M}{k} \left(\frac{1}{N}\right)^{k} \left(1 - \frac{1}{N}\right)^{M-k} \tag{4.1}$$

This is a form of the binomial distribution. It can be approximated in different ways depending on the frequency of defects: rare, occasional or frequent. The last two cases are of practical interest since, in any large scale circuit, defects are likely to occur.

1) Occasional defects. If the yield is moderate then equation 4.1 can be approximated by a Poisson distribution [Ross76]

$$\Pr(z=k) = \frac{s_0^k}{k!}\, e^{-s_0}$$

where $s_0 = \frac{M}{N}$ is the expected value of the random variable z.

A key advantage of the Poisson approximation is its simple extension to modeling multiple fabrication steps. Since the sum of independent Poisson random variables also follows a Poisson distribution

$$\Pr(z_1 + z_2 + \cdots + z_l = k) = \frac{s_l^k}{k!} e^{-s_l} \tag{4.2}$$

where $s_l = \lambda_1 + \lambda_2 + \cdots + \lambda_l$ with λ_i = expected value of z_i. For identically distributed z_i,

$$\Pr(z_1 + z_2 + \cdots + z_l = k) = \frac{(ls_0)^k}{k!} e^{-ls_0} \tag{4.3}$$

with s_0 = expected value of z_i. This contrasts with the more complex sum of geometric random variables distribution encountered in the Price model (see section 2.3).

Note that in equation 4.2 it is not necessary to assume (as in the Price model) that each processing step contributes equally to the probability of occurrence of defects. All that is necessary is to sum the expected number of defects in each *processing* step and use the sum as the parameter in a Poisson distribution. In contrast, the Price model without this assumption becomes unwieldy. Equation 2.1 becomes

$$Y = \prod_{i=1}^{4} \frac{1}{(1+d_i)}$$

where d_i is the expected number of fatal defects introduced by the i th processing step.

2) Frequent defects. For a low yield and M large, equation 4.1 is more accurately approximated by a Gaussian distribution [Ross76]

$$\Pr(z=k) = \frac{1}{\sqrt{2\pi}\,\sigma_z} \exp\left\{ -\frac{1}{2} \left[\frac{k-s_0}{\sigma_z} \right]^2 \right\} \tag{4.3}$$

where $\sigma_z^2 = s_0(1 - \frac{1}{N})$ is the variance of z.

How much more accurate is the Gaussian approximation for low but still realistic yields? First, assume N is large so $\sigma_z^2 \approx s_0$ and equation 4.3 becomes 20 is clearly lower bound on the number of chips per wafer. For n = 20, $\sigma_z^2 = s_0(1 - \frac{1}{20}) = 0.95s_0$ and $\sigma_1 = 0.98s_0$ so this approximation is highly accurate.

$$\Pr(z=k) = \frac{1}{\sqrt{2\pi s_0}} \exp\left\{-\frac{1}{2}\left[\frac{k-s_0}{\sqrt{s_0}}\right]^2\right\}$$

To compute the yield we take k = 0

$$Y = \Pr(z=0) = \frac{1}{\sqrt{2\pi}} s_0^{-1/2} \quad e^{-1/2 \ s_0} \tag{4.4}$$

Table 2.4.1 compares yield vs. s_0 for the Gaussian and Poisson approximations. With low yields (<5%), for a given value of s_0, the Gaussian approximation predicts a higher yield than the Poisson model. Since the Poisson approximation is known to underestimate yields [Glas79], the Gaussian approximation is indeed more accurate. However, the difference between the approximations is not large (~22%) even at extremely low yields (1%). The relationship between yield and area is

$$Y = \Pr(z=0) = \frac{1}{\sqrt{2\pi}} (As_0)^{-1/2} \quad e^{-1/2 \ (As_0)} \tag{4.5}$$

where s_0 = 1.202 defects per unit are per step which is derived by solving equation 4.4 for s_0 with $Y = 0.20$. A 1% yield corresponds to As_0 = 5.641 or A = 4.7 unit areas which is larger than will be considered for a CHIP building

Table 2.4.1 - Comparison of Gaussian and Poisson Approximations

Yield	s_0 Gaussian	s_0 Poisson	Gaussian/Poisson
0.01	5.64	4.61	1.223
0.02	4.49	3.91	1.148
0.03	3.83	3.51	1.091
0.04	3.38	3.22	1.050
0.05	3.04	3.00	1.013
0.06	2.77	2.81	0.961
0.07	2.55	2.66	0.962
0.10	2.05	2.30	0.891
0.15	1.53	1.90	0.805

block. Consequently, in the range of chip areas under consideration, the Gaussian approximation is only marginally more accurate than the Poisson approximation so it will not be used. The Gaussian approximation will not be further considered.

b) Indistinguishable Defects

Assume all the defects are identical and can not be told apart. With M indistinguishable defects on a wafer of N chips, there are

$$\binom{N+M-1}{M}$$

different ways of distributing the defects on the chips. To evaluate Pr(z=k), the probability that one specific chip contains exactly k defects, note that a subset of k indistinguishable defects can be chosen in only one way. The remaining M - k defects can be placed on the other N - 1 chips in

$$\binom{N+M-k-2}{M-k}$$

different ways. Hence

$$\Pr(z=k) = \frac{\binom{N+M-k-2}{M-k}}{\binom{N+M-1}{m}}$$

for small values of k and large, increasing values of N, Pr(z=k) asymtotically approaches [Glas79, Parz60]

$$\Pr(z=k;\lambda) = p\,(1-p)^k$$

$$\text{with } p = \frac{1}{1+\lambda s_0}$$

Thus a geometric distribution characterizes the defect distribution for a single processing step with indistinguishable defects.

Extending this result to multiple classes of defects, we assume that defects within each class are indistinguishable but two defects in different classes can be told apart. A different defect class is associated with each interconnection level. Since the fabrication steps are assumed to be independent, the total number of defects is the sum of the number of defects introduced by each step. By the assumption of equal defect densities at each level, the z_i are identically distributed. Consequently, Z, the total number of defects, is the sum of independent, identically distributed geometric random variables, and the probability density functions, Pr(Z=m), for 3 and 4 classes of defects are:

$$\Pr(z_1 + z_2 + z_3 = m) = \frac{1}{2}(m+1)\,(m+2)\,p^3 q^m$$

$$\Pr(z_1 + z_2 + z_3 + z_4 = m) = \frac{1}{6}(m+1)\,(m+2)\,(m+3)\,p^4 q^m$$

with $p = \dfrac{1}{1 + As_0}$ and $q = 1 - p$.

Graphs of Pr(Z=m; A) for the Poisson, 3 and 4 class models are shown in Figures 2.4.2 - 2.4.4 for different areas.

Comparing the Poisson and Price models, we find that the Poisson model is less accurate as the chip area increases. At unit area, the number of defects is overestimated. But for larger areas, the Poisson model underestimates the number of defects by a considerable amount. In short, the Poisson model is accurate only near unit area and for $m \geq 2$. Since the

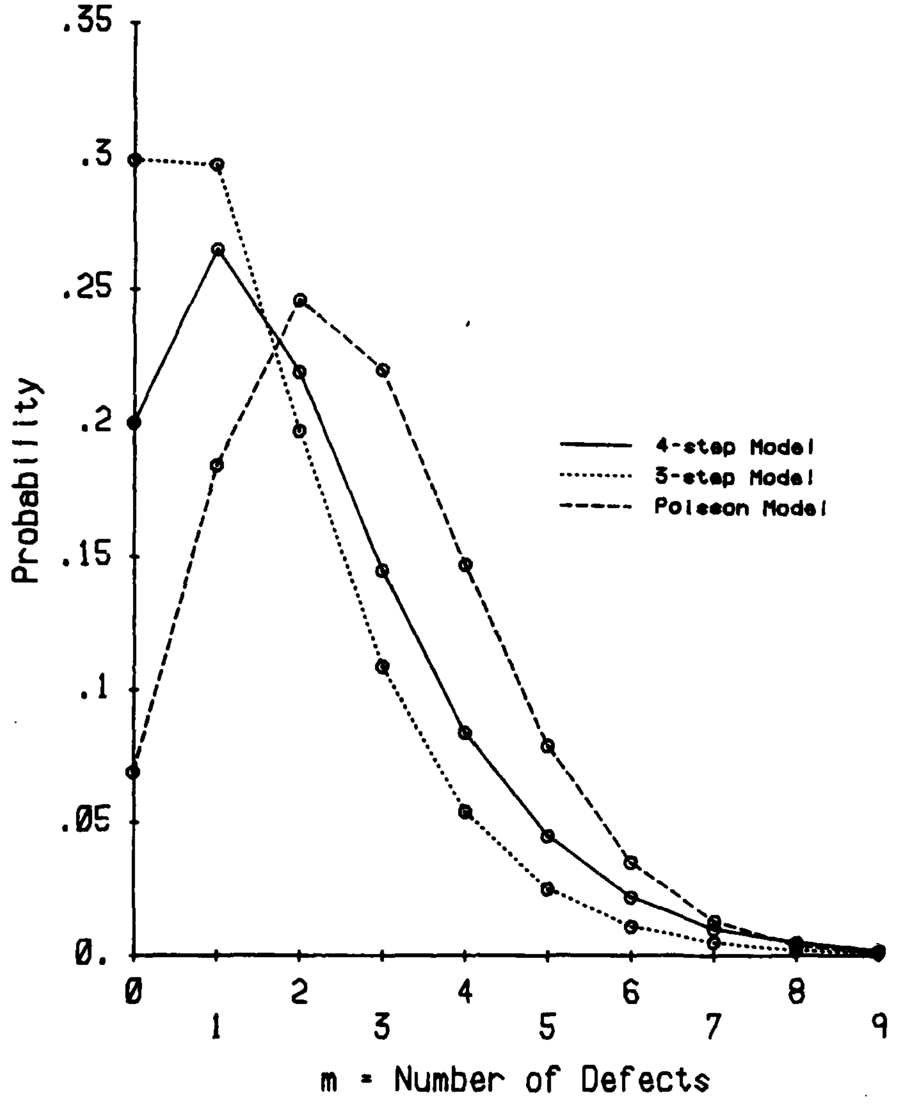


Figure 2.4.2 - Probability of m Defects
(Area = 1.0 NUA)

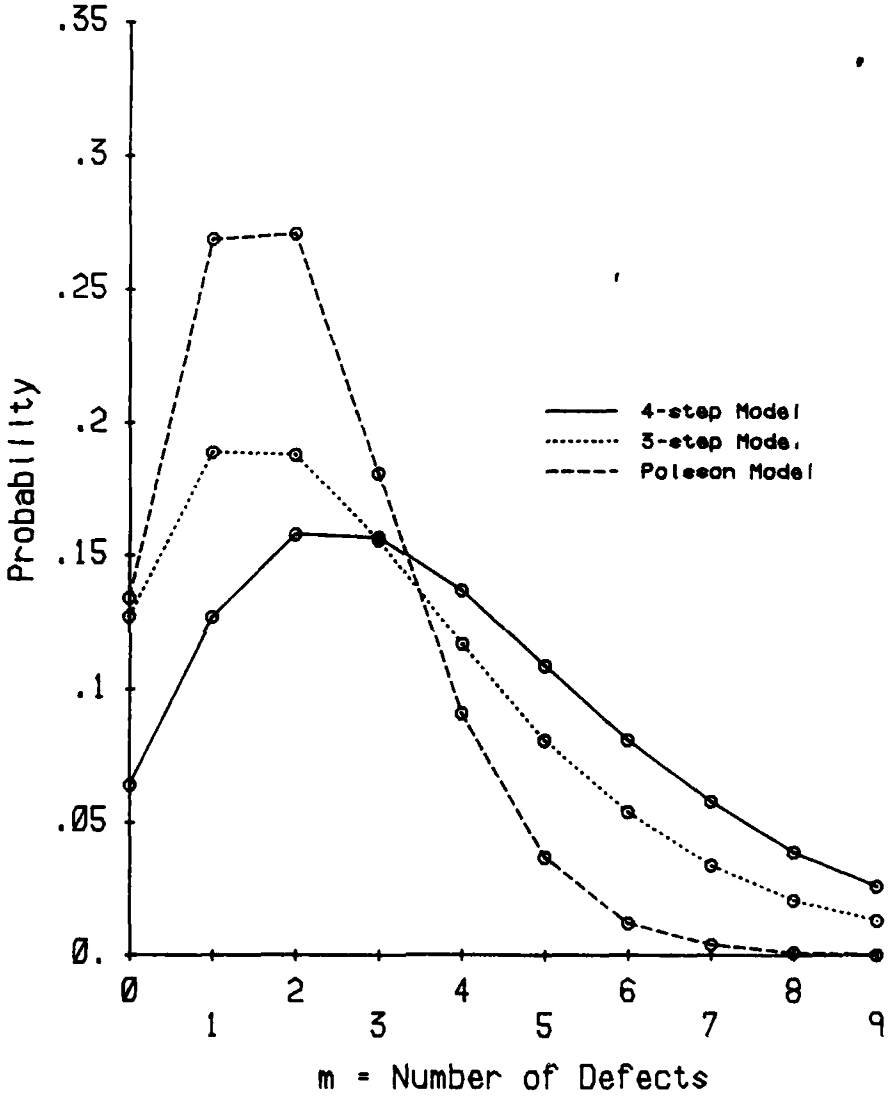


Figure 2.4.3 - Probability of m Defects
(Area = 2.0 NUA)

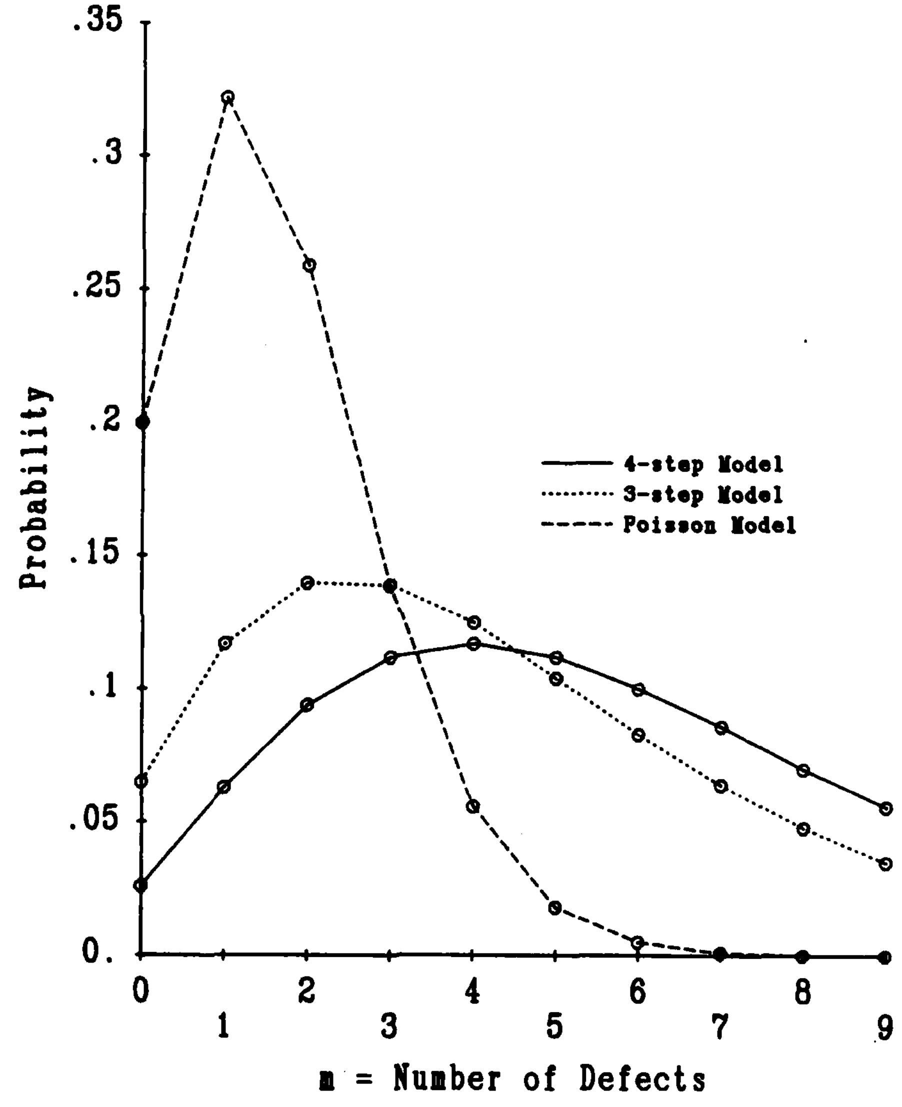


Figure 2.4.4 - Probability of m Defects
(Area = 3.0 NUA)

area of a wafer scale building block is large, and we would rather make conservative estimates than overly optimistic ones, the Poisson model is unsuitable for precise defect analysis. It is useful only for order of magnitude estimates.

Comparing the 3 and 4 class Price model, we find that both curves have very similar shapes. Furthermore, they converge as n $\rightarrow \infty$, but the 4 class model shows greater variance. The three class model is only a moderately good approximation to the four class approximation. Since a 4 level process is most appropriate for the implementation of wafer scale CHiP machines, its added complexity will be endured except when it is prohibitively costly.

5. Applications of the Yield Model

In the previous sections we developed a model of the integrated circuit manufacturing process. The analysis was based on the properties of the fabrication process. The end result was to characterize the distribution of imperfections in the fabrication process, and from this model the yield of a given size chip can be predicted.

This is not, however, our ultimate objective. In this work we are interested in the analysis of parallel processors. But the processors under consideration are fabricated out of silicon with several PEs per chip. So the modeling of integrated circuit fabrication technology is a necessary prerequisite to parallel processor analysis. The choice of the number of processing elements per chip, size of the PEs, etc. depends in part on the technology out of which the PEs are created.

In this section, the yield model developed above is applied to the study of the design of parallel processors. In very large and complex parallel processing systems, fault tolerance is a desirable (if not mandatory) property of the system. With the homogeneous structure of CHiP machines, redundancy is a natural means of achieving fault tolerance. To analyze the yield of fault tolerant CHiP modules, one must know for a chip containing a fixed number of redundant components, what is the probability that the number of faulty components does not exceed the number of redundant ones. This is the yield of the fault tolerant chip. Conversely, a design oriented version of the above question is how much redundancy is required to achieve a given yield. Knowledge of this can guide the designer of a parallel processor in choosing the amount of redundancy within the processor.

Furthermore, changes in technology impact the design of parallel processors. The scaling down of device dimensions increases yield with resulting reduction in cost. Alternatively, scaling can be exploited by using more powerful and faster PEs on a chip with the same yield. Combinations of increased PE capacity and better yield are also possible.

There are also tradeoffs between the size of the individual PEs and the dimensions of the CHiP lattice. Which is preferable, a small number of complex PEs or a larger number of simple ones? With respect to yield, this tradeoff can be quantized through the use of the yield model. These questions and others can be quantitatively answered by the application of the yield model.

a) Recovery Analysis

Given a set of Np identical PEs fabricated on a chip of area, A, what is the probability, R_m, that at most m of the PEs are faulty? This is the *recovery problem*. R_m is the probability that at least Np - m of the PEs can be *recovered* from the chip. If the chip contains m redundant PEs, R_m is the yield of the fault tolerant chip. The chip is usable if no more than m of the PEs are faulty. Otherwise the chip does not contain a sufficient number of good PEs.

From a solution to the recovery problem, the mean number of good PEs per chip is easily calculated. The probability that a chip has exactly m defective PEs is $R_m - R_{m-1}$. The expected number of good PEs is

$$\sum_{m=0}^{N_p-1} (N_p - m)\,(R_m - R_{m-1}) \tag{5.1}$$

where $R_{-1} = 0$. This is the average yield of PEs per chip.[1]

How does a solution to the recovery problem apply to the analysis of CHiP processors? CHiP machines are composed of two types of components: switches and PEs. The recovery problem considers only faults in PEs. But it will be shown (Chapter 4) that PE faults are the dominant factor in the yield of a CHiP lattice. Switches are very small and simple. As a result, they have high yield; there are few faulty switches. On the other hand, PEs are much larger, and defects are much more likely to occur in PEs than in switches. Consequently, if the PEs of a lattice are functional then there is a very high probability that the entire lattice is functional. Analyzing the yield of PEs

[1] This probability can also be calculated from the binomial distribution. Our emphasis on fault tolerant machines makes the above viewpoint (using R_m) more useful.

provides a very good approximation to the yield of the lattice as a whole.

To solve the recovery problem, note that by the assumption that all defects are point defects, each defect will disable exactly one PE. A point defect causes localized circuit damage, so it is impossible for a point defect to span two or more PEs. Consequently, if the number of defects on the chip is less than or equal to m, no more than m PEs can be faulty. In addition, recall that defects are randomly distributed over the wafer surface. It is possible for a PE to contain multiple defects. In short, the chip may contain more than m defects but they may be clustered in m (or fewer) PEs. Thus R_m consists of two terms

$$R_m = \Pr(Z \leq m; A) + \sum_{i=m+1}^{\infty} \Pr(Z=i; A) \Pr(i \text{ defects cluster in m PEs}) \qquad (5.2)$$

The distribution of Z is known from the yield model results, and the clustering probability is derived in appendix one. Different forms of the clustering probability can be derived depending on the number of classes of defects. As seen earlier, a four class assumption is the most appropriate model of the integrated circuit manufacturing process for CHiP machines. However, the solutions to the clustering probability become increasingly complex as the number of defect classes increases. Figure A1.1 in the appendix compares the solutions for one, two and three classes of defects with all defects clustering in four or fewer of 16 PEs. Note that the probability distributions converge as the number of defect classes increase. The difference between the curves for two and three classes is less than the gap between the one and two class curves. This indicates that the three and four class solutions will be in even closer agreement. Additionally, the two and three class solutions differ by only a few percent. As a result, the three class solution will be accepted as sufficiently accurate; the added complexity of the four class solution does not justify slight increase in accuracy.

Equation 5.2 gives the relationship between PE area, number of PEs, redundancy and yield. It can be used to analyze tradeoffs between these quantities. To demonstrate the results of this analysis, we will study one example that will be of considerable use in the design of the wafer scale CHiP machine. Recall that the definition of the normalized unit area is tailored to this standard PE. One NUA is defined to be the area that can hold a 2 × 2 while CHiP lattice of standard PEs with the design rules set to achieve 20% yield.

Figure 2.5.1 displays the results of applying equation 5.2 to the standard PE. On the x-axis is the number of PEs in the collection. Each one of the different curves shows the relationship between recovery probability, R_m, and the total number of PEs, N_p, for a fixed number of redundant PEs, m. Exactly m of the N_p PEs are redundant. The individual curves depict $R_0, R_1, \cdots, R_8$. This information is also displayed in Tables 2.5.1 and 2.5.2.

The lowest of the curves, R_0, is a standard yield curve. There is no redundancy so a single defect renders the chip unusable. The shape of R_0 is similar to Figure 2.2.1. Note the point $N_p = 4$ and $R_0 = .26$. One normalized unit area holds a 2 × 2 lattice and has yield .20. However, the lattice contains both switches and PEs. Some of the defects within a lattice will fall in PEs and some in switches. With the recovery curve, we are concerned only

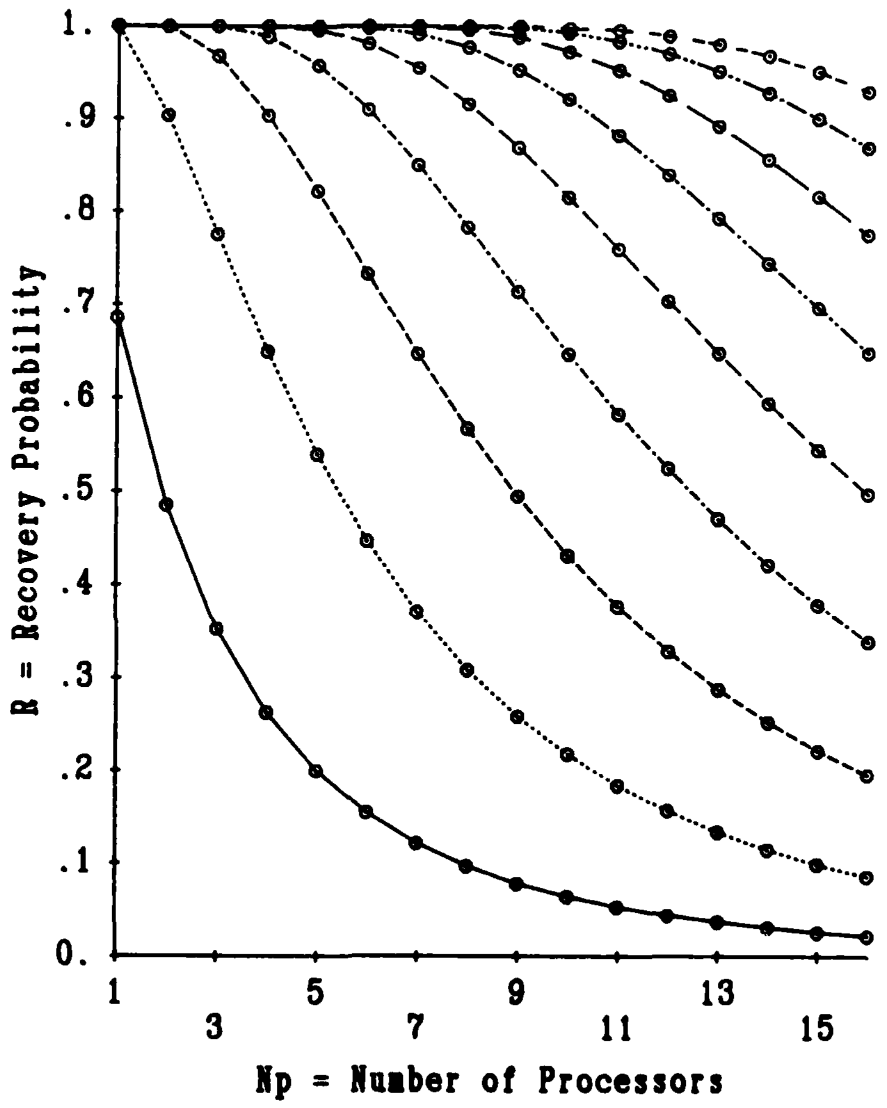


Figure 2.5.1 - Recovery Probability vs. Number of Processors

Table 2.5.1 - Recovery Probability (0-4 Redundant PEs)

	Reocovery Probability				
number of PEs	Redundant PEs 0	1	2	3	4
1	.686	1.000	1.000	1.000	1.000
2	.485	.904	1.000	1.000	1.000
3	.353	.776	.968	1.000	1.000
4	.263	.650	.904	.989	1.000
5	.200	.540	.822	.958	.996
6	.155	.447	.733	.910	.981
7	.122	.371	.647	.850	.955
8	.097	.309	.567	.783	.916
9	.078	.259	.495	.714	.869
10	.064	.218	.432	.647	.816
11	.053	.184	.377	.583	.760
12	.044	.157	.329	.525	.704
13	.037	.134	.288	.471	.648
14	.031	.115	.253	.422	.595
15	.026	.099	.222	.379	.545
16	.022	.086	.196	.340	.498

Table 2.5.2 - Recovery Probability (5-8 Redundant PEs)

	Recovery Probability			
number of PEs	Redundant PEs 5	6	7	8
1	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	1.000
3	1.000	1.000	1.000	1.000
4	1.000	1.000	1.000	1.000
5	1.000	1.000	1.000	1.000
6	.998	1.000	1.000	1.000
7	.991	.999	1.000	1.000
8	.977	.996	.999	1.000
9	.953	.988	.998	1.000
10	.922	.973	.993	.998
11	.883	.953	.984	.996
12	.840	.926	.971	.990
13	.793	.893	.952	.981
14	.745	.857	.929	.969
15	.697	.817	.901	.952
16	.649	.776	.870	.931

with the four PEs - not with the switches. Because of this the yield of four PEs is higher than the yield of a 2×2 lattice.

The size of each PE is fixed so as the number of processors, N_p, increases, the area occupied by the PEs increases proportionately. Since defects are distributed randomly, more PEs means a larger area to be "hit" by one of the defects. The R_0 decreases rapidly reflecting the fact that the yield declines as the 4th power of the area. For larger m, the decline is less steep. Redundancy moderates the effect of defects.

Figure 2.5.1 can be used in a variety of ways to analyze the design of parallel processors composed of the "standard" processing element. For example, suppose we want to produce chips containing a set number of functional PEs, but a yield higher than the R_0 curve is required. In other words, simply patterning the required number of PEs on the chip does not give high enough yield. Adding redundant PEs to the chip will increase its yield. Exactly how much redundancy is required to achieve the target yield? The answer is found in Figure 2.5.1.

For example, considering fabricating a chip that contains four good PEs. (This is not a randomly chosen example. CHiP lattices with four PEs will be used as basic units out of which wafer scale CHiP machines will be built.) Let the target yield be 75%. Simply patterning four PEs per chip results in only 26% yield (Table 2.5.1). The datapoints from Figure 2.5.1 corresponding to four PEs ($N_p = 4$ and $m = 0$; $N_p = 5$ and $m = 1$; ... $N_p = 12$ and $m = 8$) are summarized in Figure 2.5.2 and Table 2.5.3. 73% of the time four good PEs can be found in a collection of six PEs. At least four PEs are functional out of seven 78% of the time. This shows that the target yield is achieved by providing 2 - 3 redundant PEs.

From Figure 2.5.2 it can be seen that adding a single redundant PE increases recovery from 26% to 57%. This is a surprising result. Why? Adding an additional PE increases the chip area. There is more area to be "hit" by a randomly distributed defect. One might naively suppose that the addition of a redundant PE would be counterbalanced by the increase in chip area. The net result would be little or no increase in recovery. The reason this does not happen can be traced back to the characteristics of the cumulative probability distribution of the number of defects in a given area. It was noted (see section 3 - Probability Density Function) that for moderately large areas, even though there may be a large probability of at least one defect, the ***number*** of defects is likely to be small. For example, in one normalized unit area there is an 80% chance of there being at least one defect. However, the mean number of defects is less than two (Table 2.2.1). It takes only a small number of redundant PEs to absorb the few defects that are likely to occur. Thus a little redundancy provides a large increase in recovery.

b) Fault Tolerant CHiP Modules

One aspect of this work is to consider the design of CHiP modules - chips containing a small CHiP lattice. Due to pinout constraints, each module can contain only a small number of processing elements. The individual modules can be packaged and assembled to form larger CHiP machines. Alternately, the modules can remain on the wafer and be connected together to form a wafer scale machine.

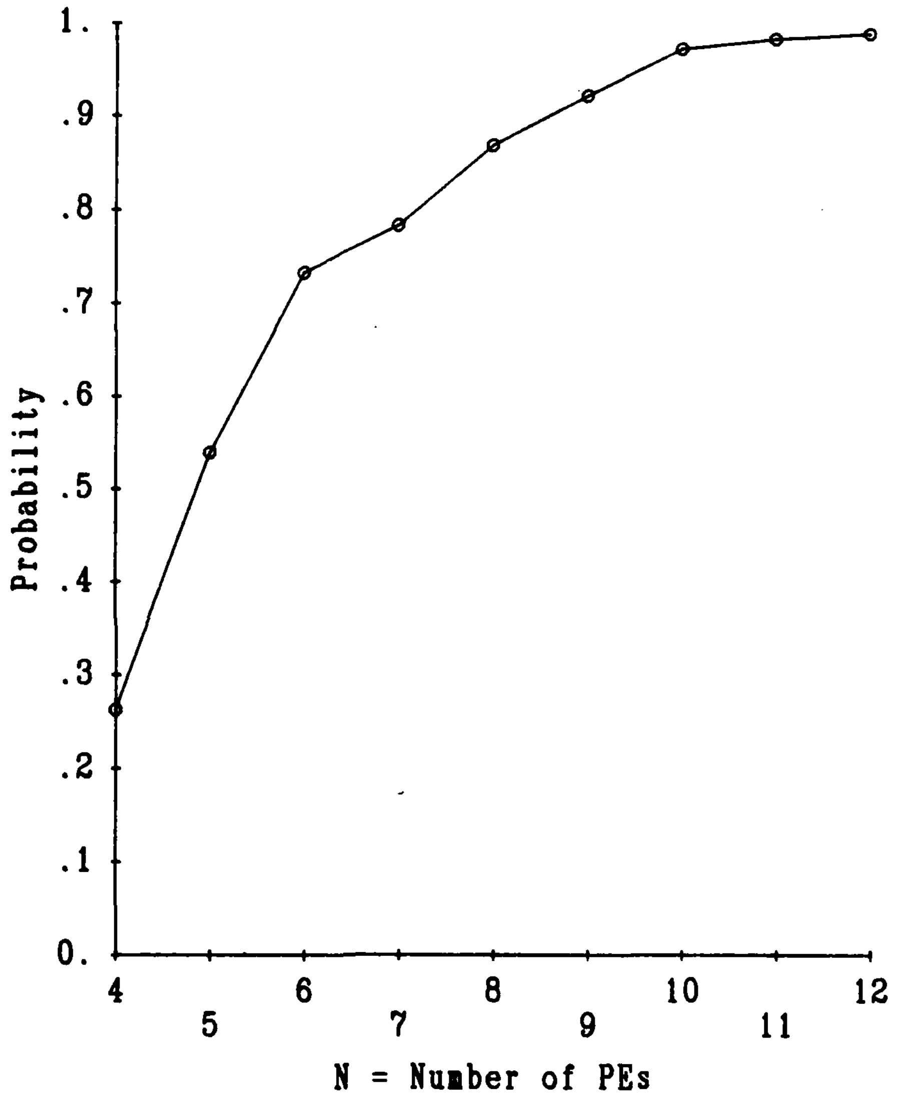


Figure 2.5.2 - Recovery Probability for Four PEs in N PEs

Table 2.5.3 - Recovery of 4 PEs from N PEs

N = number of PEs	relative area	prob ≥ 4 good PEs	number of redundant PEs
4	1.00	.263	0
5	1.25	.540	1
6	1.50	.733	2
7	1.75	.783	3
8	2.25	.869	4
9	2.25	.922	5
10	2.50	.973	6
11	2.75	.984	7
12	3.00	.990	8

The results of the previous section show that redundancy can cause large increases in yield. This suggests that redundancy could be a cost effective approach to manufacturing CHiP modules. A fault tolerant CHiP module could be designed that contains redundant PEs. The switch lattice can be used to route around the faulty PEs and connect together the functional ones. Faults, of course, can also occur in switches so redundant switches are also required.

Three problems in the design of fault tolerant CHiP modules must be solved:

- Choose the number of redundant PEs.
- Choose the switch lattice.
- Configure the lattice to avoid defects, the *mapping* problem.

The first problem can be solved using the recovery analysis results. As for the second, in Chapter 4 it will be shown that switches are quite small so they have very high yield. Doubling the corridor width of the switch lattice provides 100% switch redundancy. This allows virtually all switch faults to be absorbed. Consequently, faulty switches have virtually no effect on the yield of fault tolerant CHiP modules. The recovery analysis results (which considered only PEs) are an upper bound on the recovery of CHiP lattices containing both PEs and switches. However, this upper bound is a very close approximation to actual lattice recovery.

Finally, the lattice must be configured to mask the presence of defects. consider recovering a 2×2 white lattice (Figure 2.5.3) from a chip

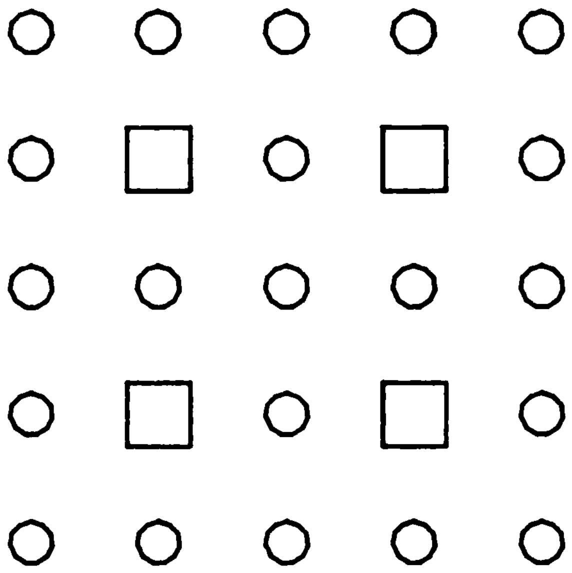


Figure 2.5.3 - 2 × 2 Virtual Lattice
(Datapaths Not Shown)

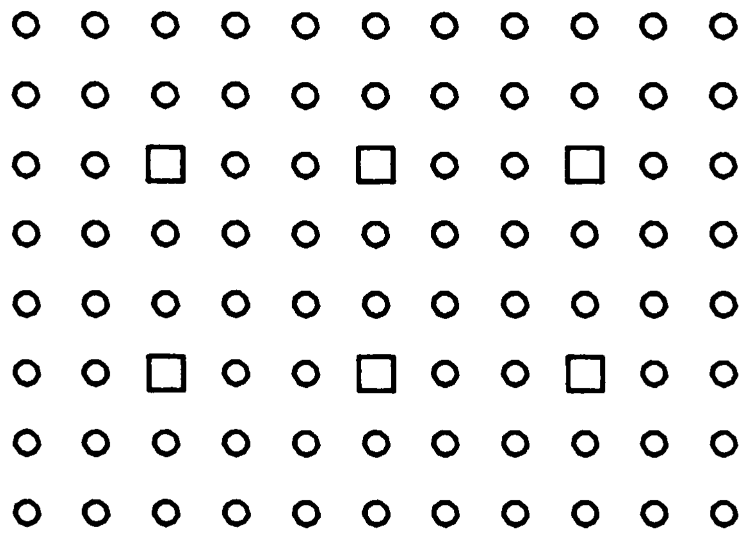


Figure 2.5.4 - 3 × 2 Physical Lattice
(Datapaths Not Shown)

containing a 3 × 2 double corridor lattice (Figure 2.5.4). The 3 × 2 lattice that is actually patterned in silicon is termed the ***physical lattice***. Switches in the physical lattice will be set so that it emulates a fault free 2 × 2 lattice, the ***virtual lattice***. We say that the virtual lattice is ***mapped*** into the physical lattice. The configured physical lattice could be used in place of the virtual lattice or vice versa. An observer of the input / output behavior of a fault tolerant CHiP module can not detect the presence or location of the faulty components.

There are two subtasks in finding a mapping of the virtual lattice into the physical lattice:

- Assign PEs and switches in the physical lattice to their counterparts in the virtual lattice.
- Define a one-to-one correspondence between datapaths in the virtual lattice and paths in the physical lattice.

The process will be explained through the example of mapping a 2 × 2 virtual lattice into a 3 × 2 physical lattice (Figures 2.5.3 and 2.5.4). The four PEs of the virtual lattice can be assigned to functional PEs in the physical lattice as shown in Figure 2.5.5. The 12 switches of the virtual lattice that are connected to ports (shaded in Figure 2.5.5a) can be assigned as in Figure 2.5.5b. The datapaths between a port and a switch in the virtual lattice become paths in the physical lattice as shown. The right port of PE A is separated from its switch by six intervening switches. The complete mapping is shown in Figure 2.5.6.

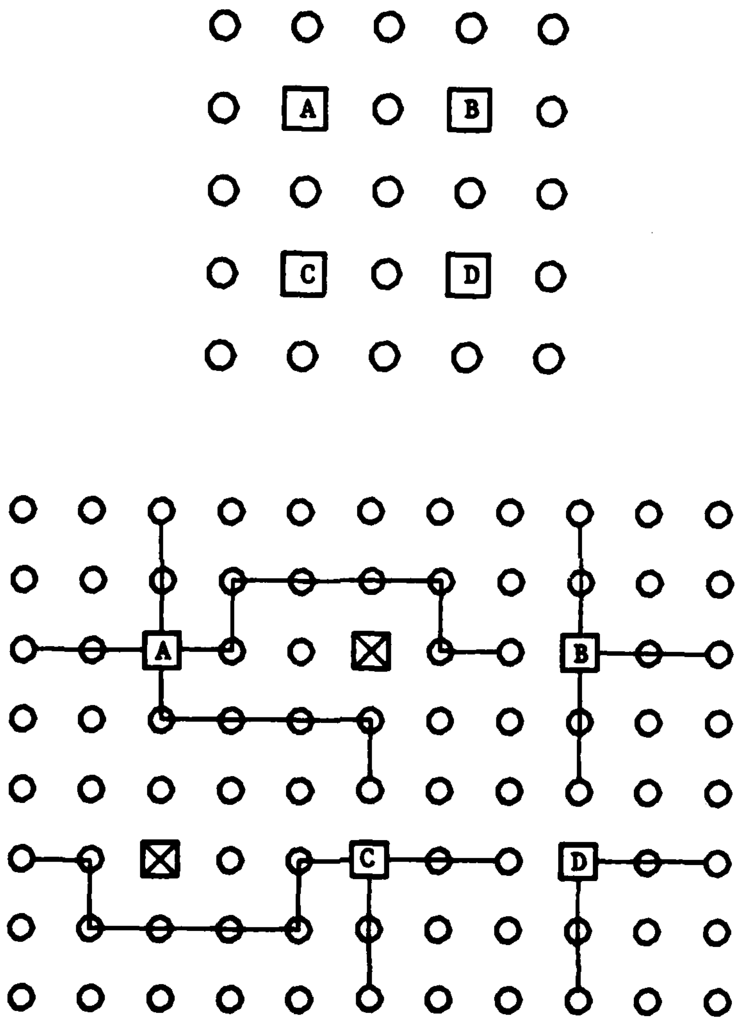


Figure 2.5.5 - Example of a Partial Mapping

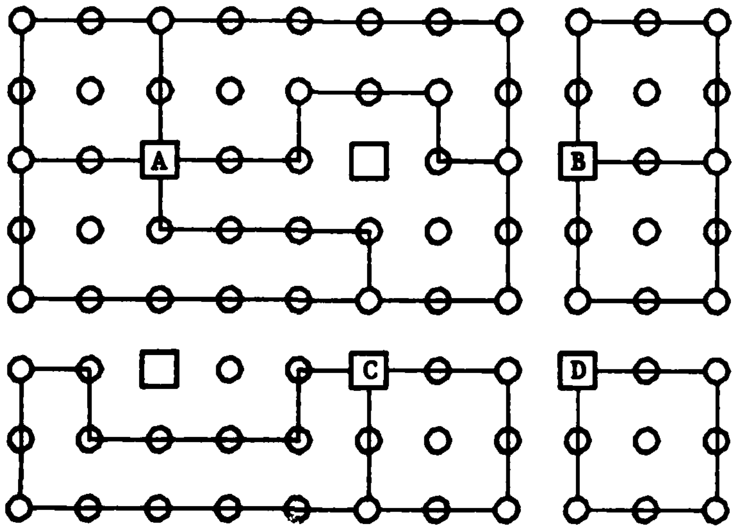


Figure 2.5.6 - Complete Mapping of the Virtual Lattice Into the Physical Lattice

c) Optimum Lattice Size

An examination of Table 2.5.3 shows that chip yield approaches one as the number of redundant PEs increases. Arbitrarily high yield can be achieved by providing enough extra PEs. However, with more PEs per chip the area of the chip increases. With larger area, fewer chips can be fabricated on a single wafer. Since the cost of processing a wafer is independent of the number of chips it holds, fewer chips per wafer leads to higher cost per chip. Unless the gain in recovery makes up for the area increase, redundancy could result in higher chip cost.

What is the level of redundancy that optimizes the number of good chips per wafer? Consider once again recovering four PEs from a chip. Using the terminology of recovery analysis, let there be Np PEs per chip. Np - 4 of these are redundant, and R_{Np-4} is the yield of the fault tolerant chips. The number of chips per wafer is proportional to the chip area. Since PEs are of fixed size, area increases linearly with the number of PEs. Hence, the number of chips per wafer is proportional to 1 / Np. Consequently, maximizing R_{Np-4} / Np determines the value of Np that also maximizes the number of good chips per wafer. In fact, 4 R_{Np-4} / Np is the fraction of PEs on the wafer that are actually used. R_{Np-4} of the chips are good. On these good chips, 4 / Np of the PEs are used. 4 R_{Np-4} / Np is the PE *utilization*.

Table 2.5.4 shows the PE utilization for the recovery of four PEs from a chip containing Np PEs. With Np = 4, 100% of the PEs on good chips are used but only R_0 = 26.3% of the chips are good. Adding one redundant PE more

Table 2.5.4 - Optimum Lattice Size for the Recovery of Four PEs

Recovery of 4 PEs			
Np = number of PEs / chip	R (%)	4 R / Np	Gain with FT (%)
4	26.3	.263	0.0
5	56.8	.456	73.3
6	77.2	.516	95.6
7	88.5	.504	92.1
8	91.6	.460	46.0

Table 2.5.5 - Optimum Lattice Size to Maximize Number of Good Chips Per Wafer

PEs Recovered (Nv)	Optimum Lattice Size (Np)	Redundancy (%)	R (%)	Gain with FT (%)
1	1	0.0	68.6	0.0
2	3	50.0	80.4	10.3
3	4	33.3	68.0	44.1
4	6	50.0	77.2	95.6
5	8	60.0	78.3	144.8
6	10	66.7	81.6	215.9
7	12	71.4	84.0	301.6
8	14	75.0	85.7	404.7

than doubles chip yield. There is a 73% (= .456 / .263 - 1) gain in PE utilization. The increase in chip yield, $R_1 - R_0$, more than makes up for the increase in chip area. With two redundant PEs, utilization increases to 96% (= .516 / .263 - 1). Adding additional redundancy reduces utilization. So Np = 6 is the optimum number of PEs per chip for maximizing the number of chips per wafer that contain four good PEs.

Why is six the optimum lattice size? The optimum is reached when the gain in recovery is exactly counterbalanced by the area increase of the chip. Examining Figure 2.5.2 it can be seen that six PEs is at the knee of the curve. Beyond this point the slope of the curve is less than one; the marginal increase in the recovery probability is less than 0.1 for each additional redundant PE. Before this point the slope exceed one; additional redundancy increases recovery by more than 0.1.

How many more good chips per wafer are there? It will be shown (Chapter 5) that a standard PE occupies a 1.75 mm × 1.75 mm region of silicon. A chip containing four PEs is therefore of size 3.5 mm × 3.5 mm. (This estimate ignores the area occupied by bonding pads and their drivers.) The number of square chips with edge length e that can be packed onto a circular wafer of diameter D is [Phis79]

$$\frac{\pi D^2}{4 e^2} - 1.77 \frac{D}{e}$$

A 4" wafer can hold 647 four PE chips. At 26.3% yield a wafer has 170 good chips. A six PE chip has 50% more area. Assume that it occupies a square with edge 3.5 $\sqrt{2}$ = 4.29 mm. A 4" holds only 399 of these larger chips. But redundancy has increased the yield to 77% resulting in 308 good chips per wafer. Thus redundancy has resulted in an additional 308 - 170 = 138 good chips per wafer - an 81% increase. The fixed cost of processing a wafer is divided between more chips. In short,

- redundancy can substantially decrease the manufacturing cost of chips containing several processing elements.

The optimum lattice size for recovering Nv PEs per chip with Nv ranging from one to eight is shown in Table 2.5.5 and Figure 2.5.7. In every case except for Nv = 1, redundancy can increase the PE utilization and subsequently reduce cost. The gains in utilization increase with Nv. This is because the baseline for the comparison (no fault tolerance) is a standard yield curve. As shown earlier, yield decreases rapidly as a function of area (Figure 2.2.1). So as Nv increases, the baseline utilization drops sharply.

Additionally, the percentage redundancy required at the optimum lattice size increases as a function of Nv. With lattices occupying a large area, a higher fraction of the PEs must be redundant. With large lattices, there is a decline in the marginal increase in redundancy of each extra PE added. More redundant PEs are required to provide the same level of protection against defects.

d) Design Analysis

By combining the yield model with recovery analysis, the interrelationships between PE size, lattice dimensions, redundancy and yield are known. Tradeoffs between these quantities can be assessed. Since the manufacturing cost of a chip depends on its yield, these results show how various factors of the parallel processor design effect its cost.

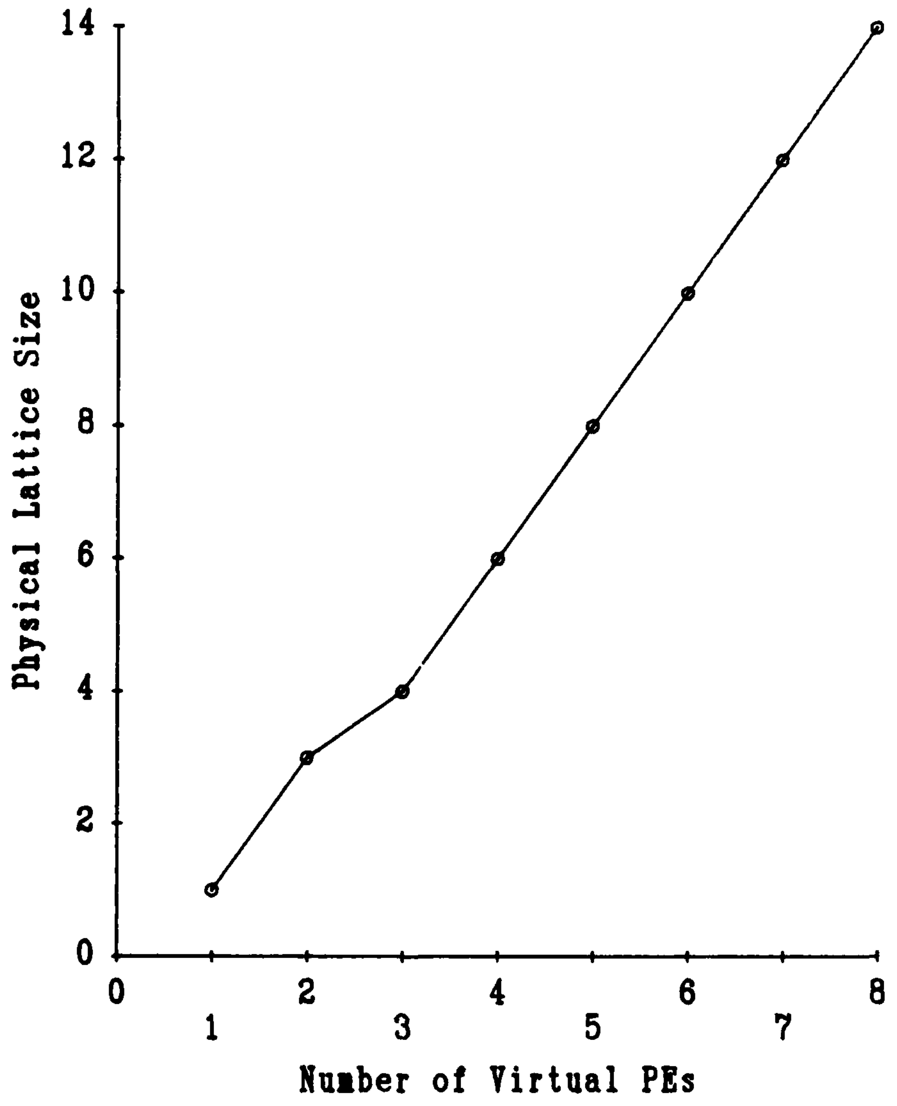


Figure 2.5.7 - Optimum Lattice Size to Maximize Number of Good Chips Per Wafer

In the previous sections, the effect of redundancy on yield was studied. However, the methodology of the yield model and recovery analysis can be used to investigate a wide variety of design tradeoffs. The primary advantage of this methodology is that it provides *quantitative* analysis. We consider one example below.

The state of the art of integrated circuit manufacturing is not static. The dimensions of individual devices continue to shrink. Given a design of a parallel processor which is constructed from chips containing several PEs, what is the effect of advances in technology on the machine? How will the yields of the individual chips improve? How much redundancy is required with smaller PEs? Figures 2.5.8 and 2.5.9 display the recovery probabilities for device area scaled by a factor of one half and one quarter respectively. We assume the same standard PE is produced only at doubled and quadrupled density.

Let us reconsider the example proposed in section A - manufacturing a chip with four good PEs at 75% yield. With device area shrunk by a factor of two, only one instead of two redundant PEs are required. The recovery of four good PEs from a set of six jumps from 75% to 95%. With quadrupled density, no redundancy is required. The yield of a chip containing four standard PEs is about 70%.

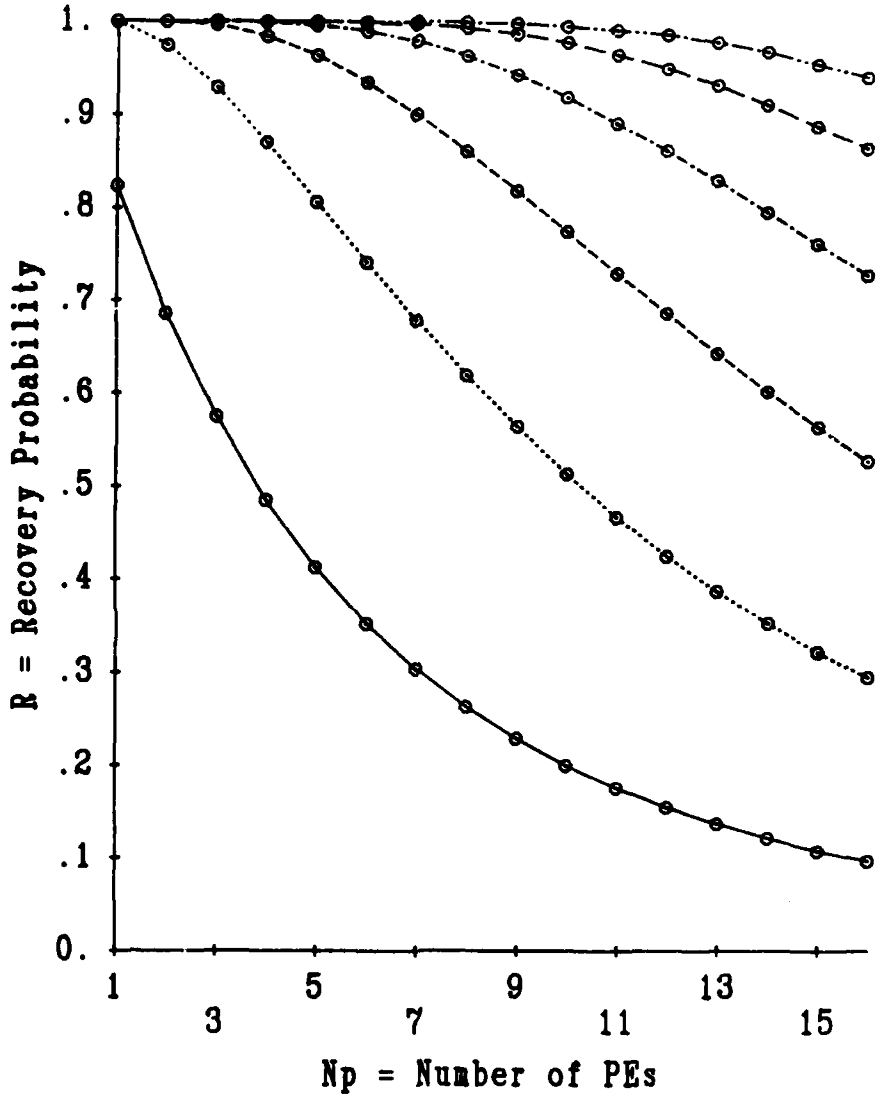


Figure 2.5.8 - Effect of Scaling on Recovery
(Scale Factor = 0.5)

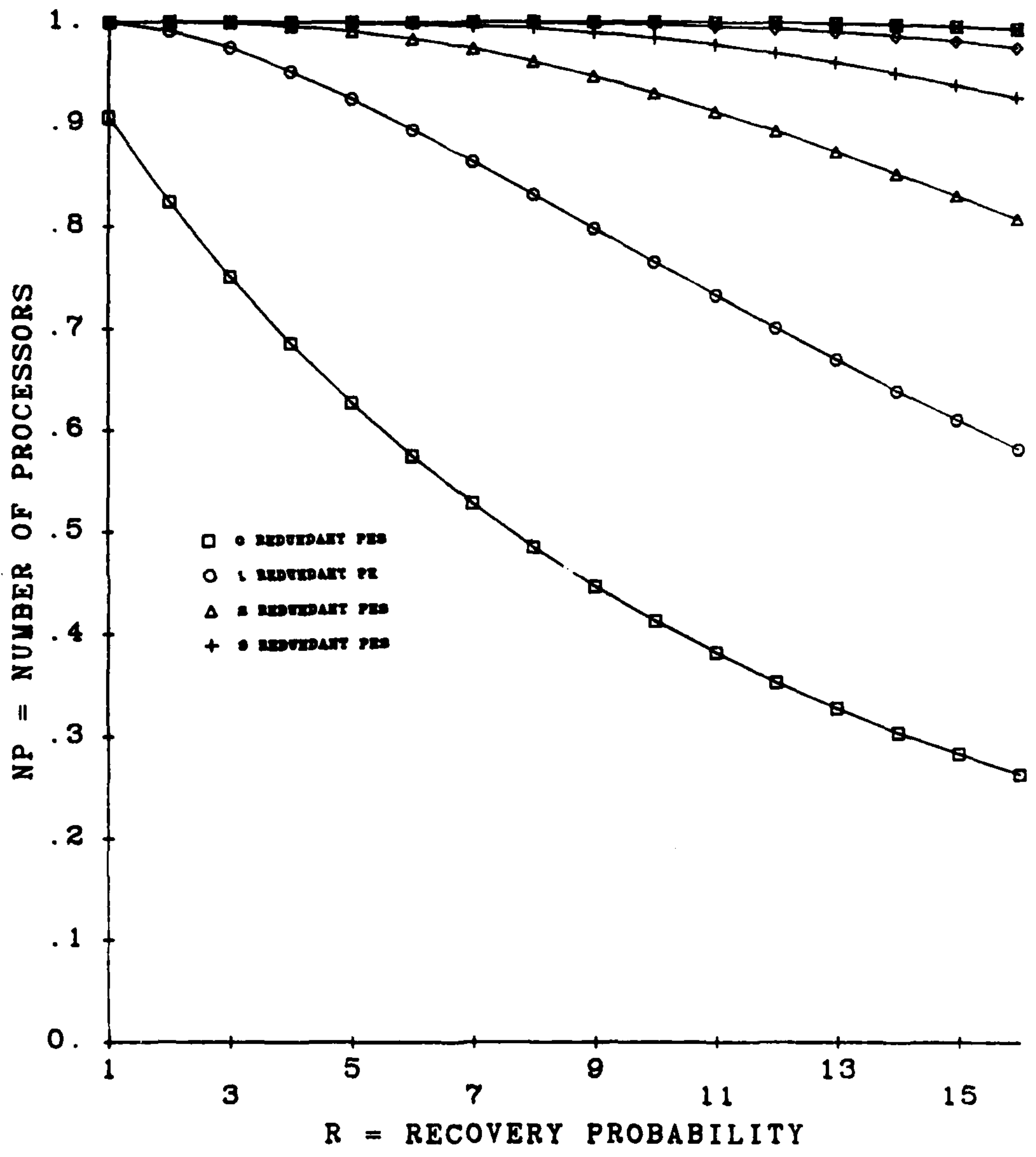


Figure 2.5.9 - Effect of Scaling on Recovery
(Scale Factor = 0.25)

Instead of exploiting the increase in density to manufacture the same design more economically, it can also be used to produce a more powerful machine at the same cost. For example, with doubled density, nine PEs per chip can be fabricated with about the same yield as four PEs per chip at the previous density. Assuming pinout constraints are satisfied, the lattice dimensions can be increased by a factor of 2.25 without increasing the number of chips in the machine and for approximately the same cost.

This methodology can be used to investigate many other tradeoffs in the design of a parallel processor. The effect of technological advances is but one such example. Many design decisions reflect themselves in terms of area or yield. This lends considerable generality to the methodology presented here.

CHAPTER 3

TWO LEVEL HIERARCHY

In this chapter, we return to the problem of designing a wafer scale CHiP processor. The goal is to fabricate a large-scale parallel processor on a single wafer of silicon. There are many problems to be considered in the design of such a system: processing element design, testing, PE to PE communication, power consumption, etc. In this section, we consider the problem of *structuring* a wafer containing individual switches and processing elements into a CHiP processor.

As shown in Chapter 1, structuring is the key problem in the implementation of any wafer scale system. Since the semiconductor manufacturing process is imperfect, each wafer contains many defective PEs and some defective switches. These must be bypassed so their presence is masked. Only the good processing elements and switches are connected together. Furthermore, the good components must be connected to form a CHiP lattice. The structured wafer emulates a smaller but fully functional CHiP lattice.

This chapter synthesizes previously presented ideas of wafer structuring by column exclusion (Chapter 1) and of fault tolerant CHiP modules (Chapter 2). A *two level decomposition* of the structuring problem is proposed. The basic idea to divide the wafer into a number of separate *building blocks*. Each building block contains sufficiently many redundant components to insure that a smaller functional lattice exists within almost every block. Virtually every block on the wafer will contribute a small subpart to the overall structure; the blocks have high yield. In addition, the switch lattice of the blocks provides a substantial amount of wiring bandwidth through the block. A very large number of independent wiring paths can pass through from one side of the block to the other.

Recall that the column exclusion strategy for structuring has two requirements: high yield and wire around capability. Redundancy within the building block insures high yield, and the switch lattice of the building block provides the wire around capability. As a result, building blocks modules are suitable for using the column exclusion strategy for wafer structuring. This makes CHiP machines a natural choice for wafer scale implementation.

Before explaining the two level decomposition further, the structuring problem and its global solution are examined. This will provide the motivation for the decomposition of the wafer into building blocks.

1. The Structuring Problem

We are given a wafer with a very large lattice patterned on it. Due to circuit defects, every wafer will contain both faulty PEs and faulty switches. It is assumed that the yield model and recovery analysis of Chapter 2 apply to the lattice, and that the lattice has been completely tested. (This is a difficult problem by itself. It is considered in detail in Chapter 6.) The status, good/bad, of every component in the lattice is known. All functional components have been found, and no dysfunctional components have been incorrectly identified as good.

The goal is to ***structure*** the wafer so it behaves as a smaller but fully functional lattice. The switch lattice is used to bypass faulty components. An observer of the input/output behavior of the structured wafer can not detect the presence, number or location of the faults. Additionally, the wafer is structured so that it emulates a virtual lattice (see Chapter 2). The behavior of the structured wafer and the virtual lattice are identical.

For example, Figure 3.1.1 shows one method of structuring a wafer. For simplicity the switches are not shown. The wafer contains a lattice of dimension 6 PEs by 5 PEs with ten of the PEs defective. A 4 × 4 virtual lattice (Figure 3.1.2) is mapped onto the wafer. The numbering of the PEs shows the correspondence between elements of the structured wafer and the virtual lattice. The logical structure of the virtual lattice and the structured wafer are the same since their components are connected in identical topologies. The structured wafer could be used in place of the virtual lattice or vice versa.

There are two subproblems to the structuring problem. The first is to specify the lattice structure that is patterned on the wafer. Secondly, an algorithm for structuring the wafer into a fault-free virtual lattice must be specified.

The designer has complete freedom in choosing the lattice parameters: PE and switch redundancy, corridor width, switch degree, crossover capability, datapath width, etc. As in the fault tolerant CHiP modules previously discussed (Chapter 2), increased wiring bandwidth must be

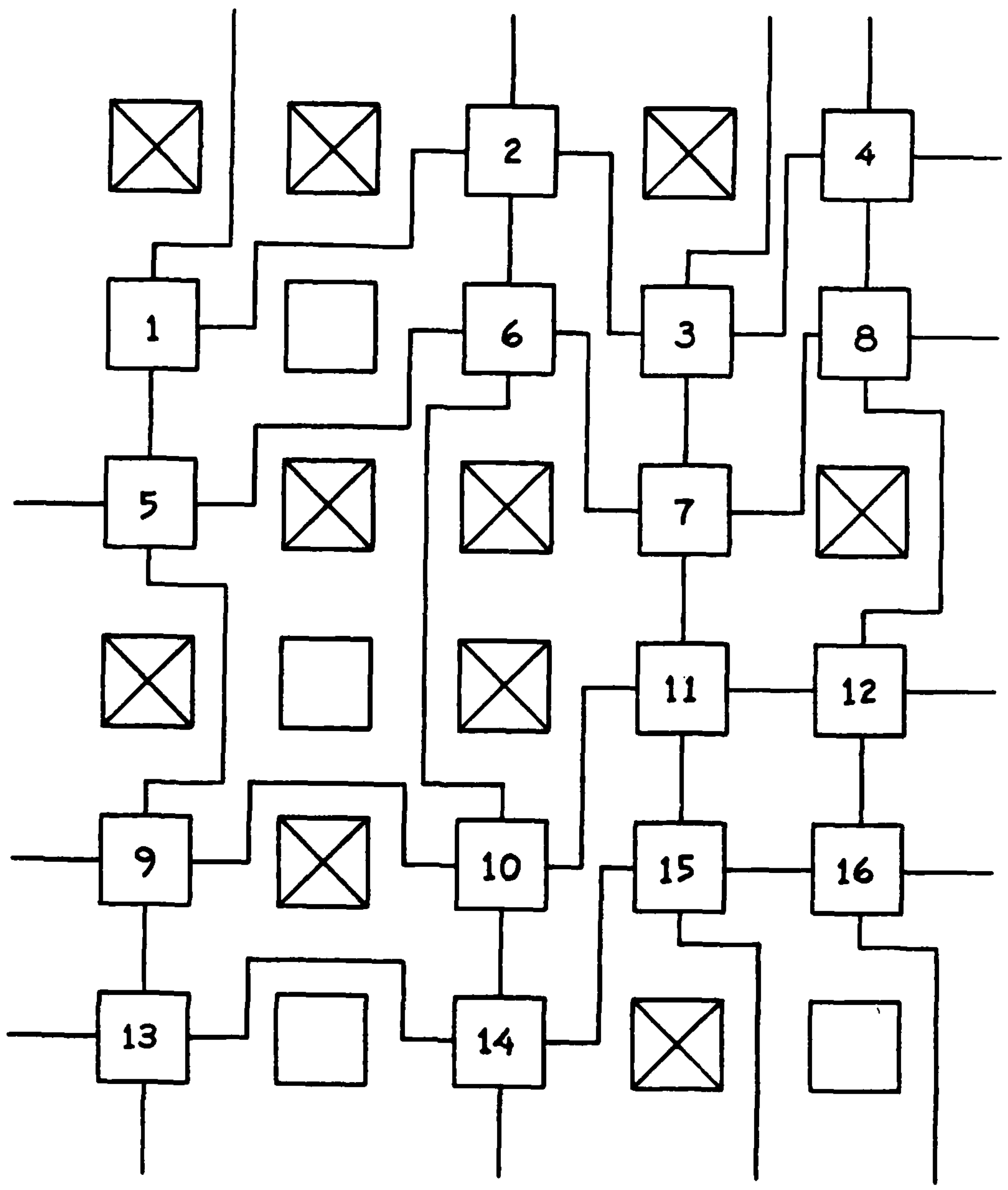


Figure 3.1.1 - Example of a Structured Wafer -
4 × 4 Virtual Lattice in a 6 × 5 Lattice

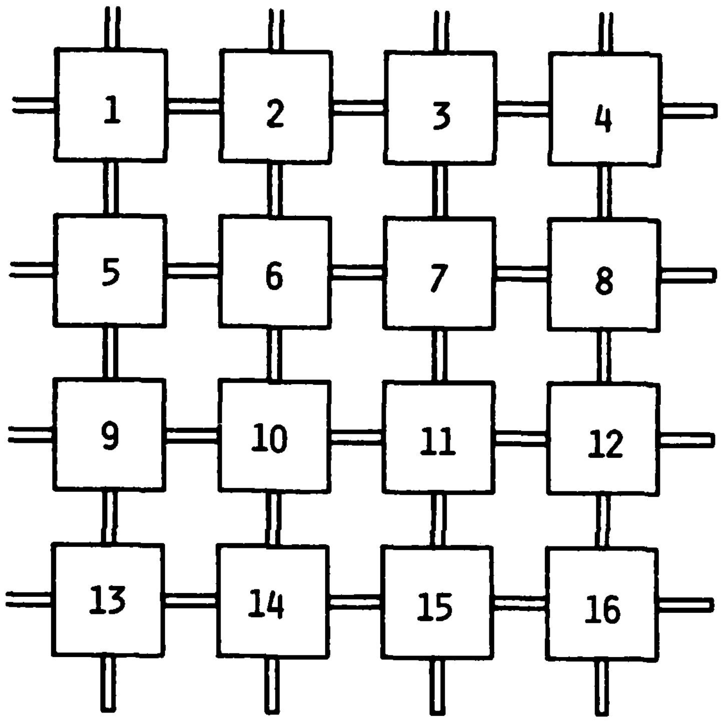


Figure 3.1.2 - 4 × 4 Virtual Lattice Which Is *Functionally Equivalent to the Structured Wafer*

provided to route around faulty components. This additional wiring capability can be implemented with a combination of extra switch corridors, additional crossover capability and increased switch degree. The goal is to provide sufficient additional wiring bandwidth to be able to replace faulty components and also to route around the defects.

The flexibility gained by the additional wiring bandwidth within the lattice is not without its cost. Extra switches or additional switch complexity are overhead that is required for fault tolerant reconfiguration. This overhead consumes wafer area which could be occupied by processing elements. Perhaps more importantly, it also adversely effects performance by increasing the number of switching levels between PEs. Every extra switch a signal must traverse introduces additional impedance and capacitance. This increases the time of flight of the signal and reduces the speed with which PEs can communicate. Consequently, one design objective is to minimize switching overhead while still insuring the reconfigurability of the wafer in the presence of faults. The choice of lattice parameters will be deferred until Chapter 4 on "Building Block Design." This chapter concentrates on the second goal.

An algorithm must be specified for performing the structuring. The input to the algorithm is the status, good/bad, of all the components on the wafer. The algorithm must compute all switch settings necessary to structure the wafer into a CHiP processor (*i.e.* the virtual lattice). There are two aspects to this problem: virtual lattice selection and mapping the virtual lattice onto the wafer. Given a wafer (with faults, of course), the dimensions of the virtual lattice to be emulated must be decided upon.

After choosing the virtual lattice size, it must be mapped onto the wafer (see Chapter 2): the virtual switches and PEs are associated with their counterparts on the wafer, and the datapaths of the virtual lattice are mapped into paths of switches. First, consider perhaps the simplest algorithm for structuring the wafer.

2. Global Strategy

In the global strategy, the wafer is considered to be a single, continuous lattice. The choice of a virtual lattice and the mapping problem are applied to the wafer as a whole. Thus the name of the approach - the algorithms are applied globally to the entire wafer. From the wafer, a single large virtual lattice is extracted, and it is mapped onto the entire wafer surface. The virtual lattice is mapped onto the wafer just as in the fault tolerant CHiP modules (Chapter 2). Figure 3.1.1 depicts an example of a global structuring.

Several problems are encountered with this approach. First, two logical neighbors in the virtual lattice are not necessarily in nearby locations on the wafer. They may be separated by long distances. This results in very long paths between PEs. Figure 3.1.1 depicts an example of this for a small lattice. A path between PEs, instead of going to an adjacent neighbor, may have to route around several intervening PEs. With the much larger lattices (*e.g.* 30 PEs by 30 PEs) that can be fabricated with current technology on a 4" wafer, very long path lengths can result. This causes serious signal propagation delays. Furthermore, due to the pipelined nature of the computations performed, a CHiP machine is no faster than its slowest link. A single long path reduces the performance of the entire machine.

As a result, it is desirable to minimize the maximum path length in a mapping. This is difficult in general to achieve for two reasons. First, the mapping problem for the whole wafer is by itself computationally difficult. Attempting a simultaneous minimization over all possible mappings is not practical. Second, even if a minimax path length mapping is obtained, there is no guarantee that it will be acceptably short. The minimax path length for the global structuring may be so long that it seriously impairs machine performance. A global solution to the structuring problem may inherently lead to unacceptably long path lengths.

Second, given the selection of a virtual lattice, consider the problem of mapping the virtual lattice onto the wafer. The number of possibilities for the mapping between the virtual lattice and lattice patterned on the wafer grows exponentially with the total number of components. Since a wafer can hold a very large lattice, exhaustive search techniques for finding a mapping are not practical.

The mapping problem is an instance of the subgraph homeomorphism problem [Gare79, LaPa78a, LaPa78b]. No known polynomial algorithm exits for the mapping problem. Furthermore, the global strategy gives rise to a very large instance of the mapping problem. A 30 PE by 30 PE double corridor lattice (which is feasible to fabricate on a single wafer - see Chapter 5) contains over 20,000 switches and PEs. Even a polynomial time algorithm may not be computationally tractable on problem instances of this magnitude.

In summary, the global approach leads to a computationally intractable structuring problem combined with potentially poor performance of the resulting CHiP processor. What is needed is a means of reducing the size of the mapping problem and placing a limit on the minimax path length of any mapping. In the following section, a divide and conquer approach, the two level decomposition, is proposed which achieves these objectives.

3. Two Level Decomposition

Rather than trying to structure the wafer as a whole, the idea of the two level decomposition is to divide the wafer into logical pieces. A virtual lattice is mapped into each of these pieces, and the individual solutions are composed to form a larger CHiP lattice. The organization of the wafer is divided into two components: the individual pieces and their composition which forms the wafer scale CHiP processor. There is a *two level hierarchy* within the processor - the individual pieces are the components out of which the wafer scale machine is built. This division of the problem into small pieces leads to a computationally tractable divide and conquer approach to the structuring problem.

Each of the individual pieces is a *building block* of the wafer scale machine. From each block we will extract a lattice of fixed size. For the blocks proposed in the following chapter, a 2×2 lattice is extracted. This eliminates the problem of choosing the dimensions of the virtual lattice (at the cost of sometimes underutilizing the good components of the block). All blocks yield the same size lattice regardless of how many functional PEs and switches they contain. More importantly, the uniformity of the virtual lattice size makes it easy to compose the individual lattices. Each block contributes a fixed size piece to the overall machine. Each of the pieces connects to its four neighbors in a simple and regular manner (Figure 3.3.1).

In contrast, if blocks contribute virtual lattices of different sizes (see Figure 3.3.2), this introduces difficult problems of matching the pieces. Simplicity is a key to success.

Figure 3.3.3 depicts an example of structuring with a two level hierarchy. The faulty or simply unused processing elements are marked with Xs. A 6 × 4 lattice is patterned on the wafer. (For simplicity, switches are not shown. The structuring of the switches is performed similarly to the structuring of the PEs.) In the first level of the hierarchy, the wafer is divided into four building blocks each containing a 3 × 2 lattice. A 2 × 2 virtual lattice is mapped into each of these blocks. The individual 2 × 2 lattices are in turn connected together to form a 4 × 4 array of processors on the wafer surface. The structured wafer is functionally equivalent to the 4 × 4 lattice in Figure 3.1.2.

In this particular example, no building block has more than two faulty processing elements so a virtual lattice can be mapped into every block. In practice, some blocks may not contain enough functional components to host a virtual lattice - the block is considered faulty. The random nature of defects makes it impossible to completely safeguard against this possibility. The column exclusion strategy is used to deal with faulty blocks. Wherever a faulty block occurs, the entire column (or row) containing that block is excluded. In order to efficiently implement column exclusion, blocks must have high yield and wire around capability. These problems are discussed in Chapter 4 on Building Block Design.

The advantages of the two level composition are twofold. First, a bound is placed on the maximum path length in the lattice. The mappings

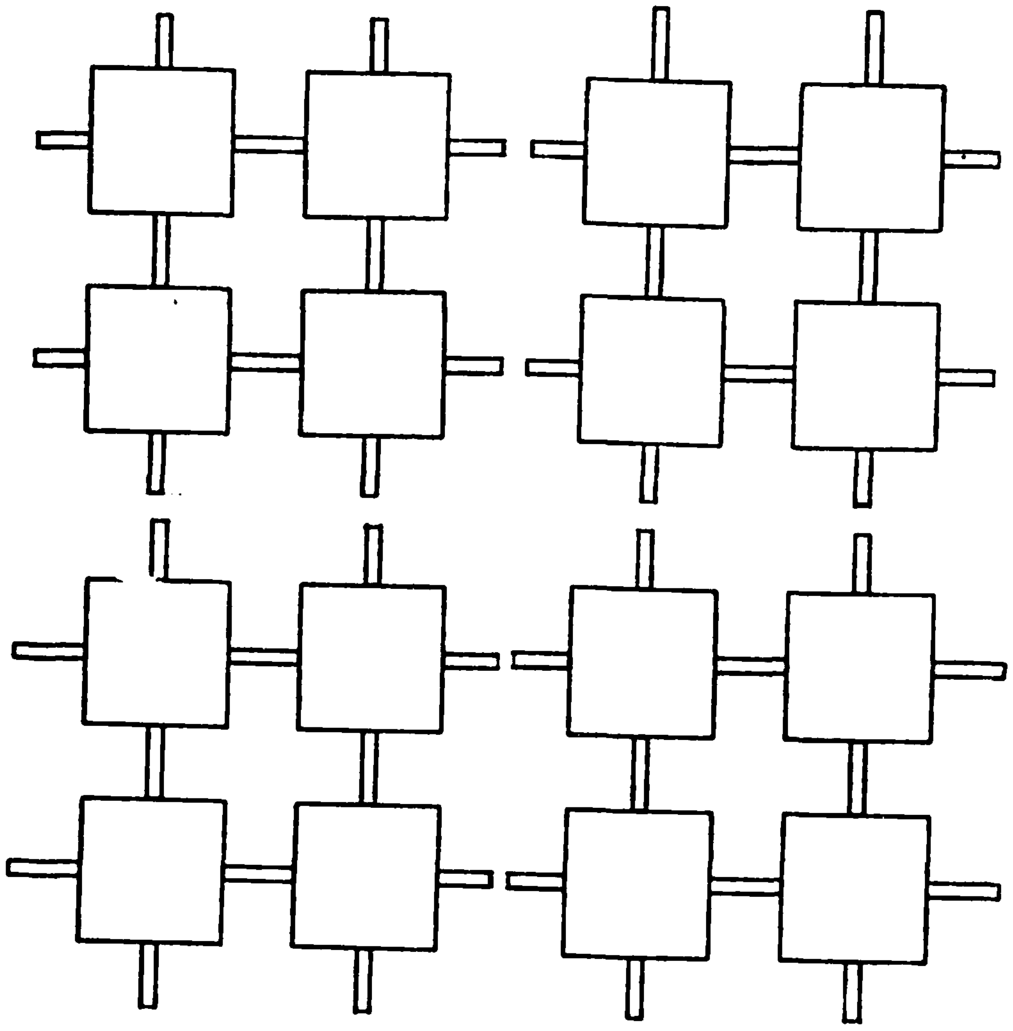


Figure 3.3.1 - Composition of Lattices of Identical Size

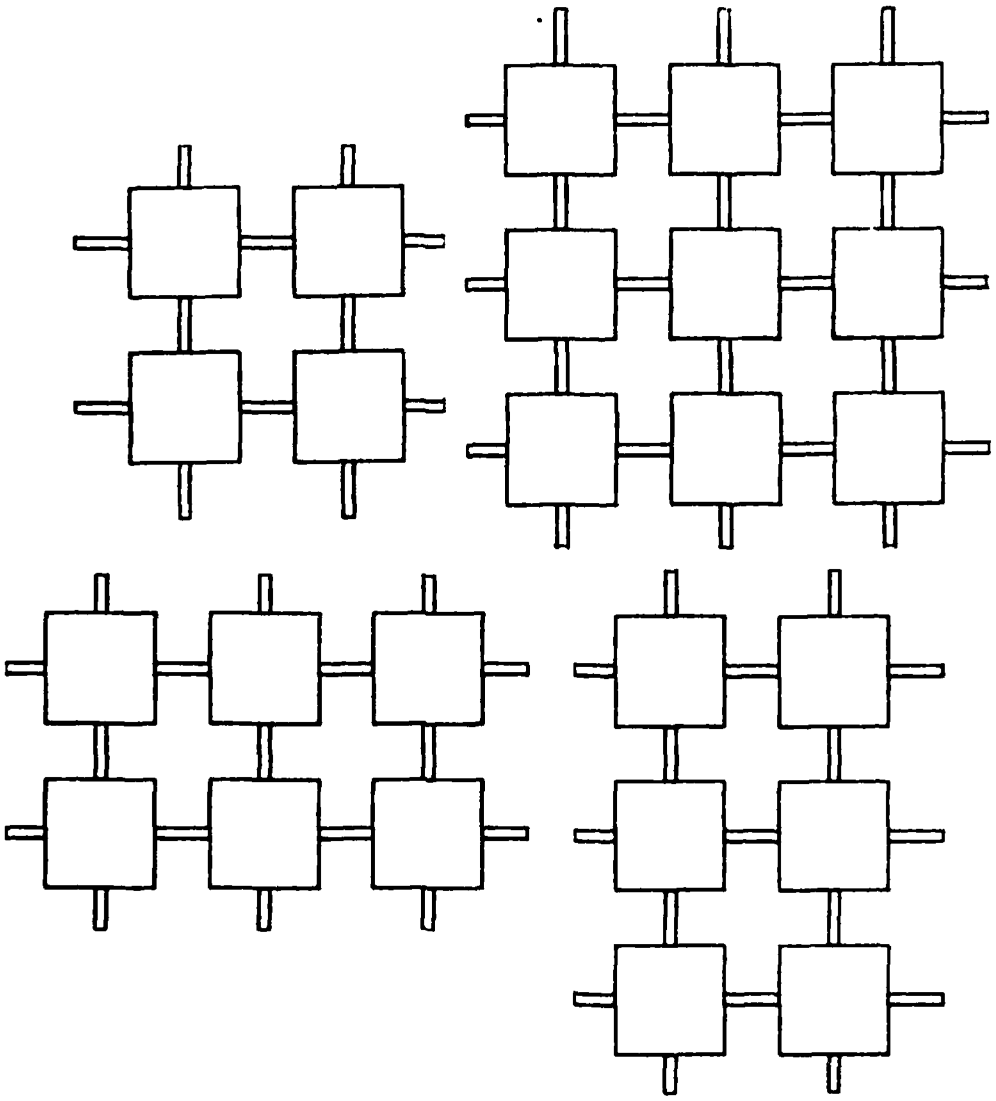


Figure 3.3.2 - Composition of Lattices of Nonuniform Size

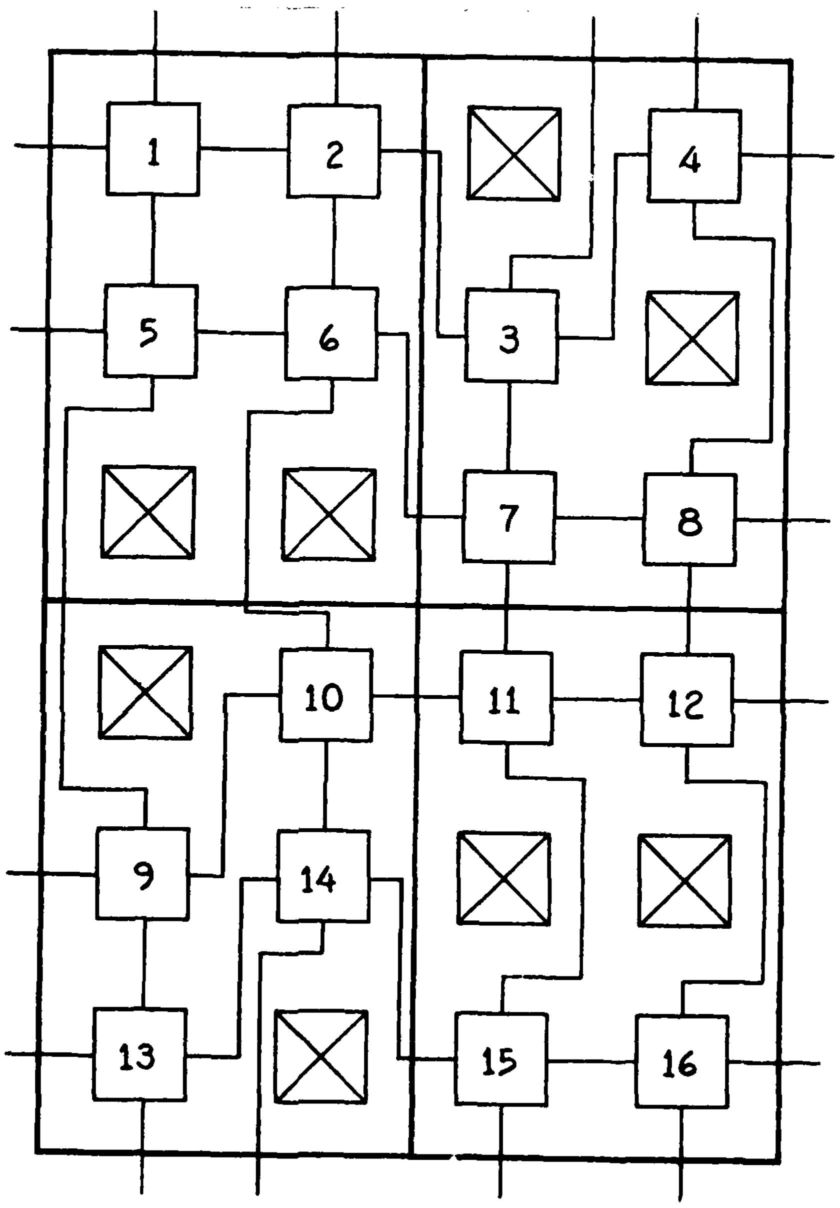
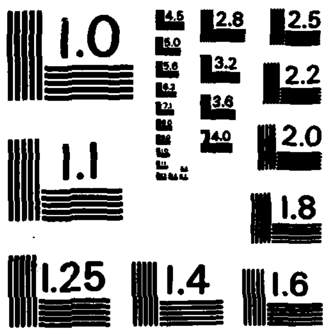


Figure 3.3.3 - Structuring With the Two Level Hierarchy

AD-A121 886 WAFER SCALE INTEGRATION OF PARALLEL PROCESSORS(U) 2/3
PURDUE UNIV LAFAYETTE IN DEPT OF COMPUTER SCIENCES
K S HEDLUND NOV 82 CSD-TR-411 N00014-80-K-0816
UNCLASSIFIED F/G 9/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0

performed on the individual blocks are contained totally within the block. Any two PEs in the virtual lattice mapped into a block are connected by a path which does not go outside the block. This limits the maximum length of any path and establishes an upper bound on the processor to processor communication time.

Second, the problem of structuring the wafer is made computationally tractable. The one very large instance of the mapping problem that is generate by the global strategy is divided into many small instances. Each of the building blocks is small, and the virtual lattice can be mapped onto it by brute force methods. Since the same size virtual lattice is mapped into each block, individual solutions are easily composed. In short, the structuring problem is made computationally tractable by a divide and conquer approach.

The primary disadvantage of the two level decomposition is that fewer good PEs are used than in the global strategy. By extracting a fixed size lattice from each block there will be functional but unused PEs on the wafer. Many of the blocks on the wafer will have more good PEs than are used in the virtual lattice. These extra PEs will not be utilized now. Additionally, no PEs in the excluded columns are used.

Area is clearly sacrificed in the two level hierarchy. But the commodity in greatest supply in a wafer scale system is area. The two level hierarchy trades area for performance and simplicity of structuring.

Additionally, the good but unused PEs can be held in reserve for future use. During the lifetime of the wafer scale CHiP processor, if a PE fails, an unused PE can be switched in to take its place (see section 7.5a). This requires only a local modification to the affected building block. Thus even after manufacturing is complete, the wafer scale CHiP processor has considerable fault tolerance.

CHAPTER 4

BUILDING BLOCK DESIGN

This section considers the design of a building block of a wafer scale processor. A building block implements the first level of the two level hierarchy. Each functional block is configured into a virtual lattice. This mapping is performed as with fault tolerant CHiP modules (see section 2.5b). The wafer has patterned on it a grid of blocks typically 8 × 8 to 10 × 10 in size which is structured by column exclusion - wherever there is a faulty block, the entire column containing that block is excluded from the grid. To be practical, the column exclusion strategy has two requirements: high block yield and the capability to wire around unused columns of blocks. These requirements are examined in detail and a quantitative evaluation is made.

Several important design choices must be made for building blocks. In order to provide high block yield necessary for column exclusion, fault tolerance is an essential characteristic of the building block. The amount of redundancy within a block is one of the major design choices, and it is dependent on the yield of the individual processing elements. Since yield is directly related to area, the size of the CHiP processing elements must be known. To estimate their area, the intended primary application of CHiP processors, systolic algorithms, dictates the minimum functional requirements of a processing element. From this, a high level floor plan of a processing element is proposed. The floor plan combined with the sizes of individual register, ALU and control cells gives a rough estimate of the area of the processing element without actually designing the PE in detail.

Once the area of a PE is known, our previously developed technique of recovery analysis is used to determine the lattice dimensions of a building block. After a similar consideration of switch design and estimating switch yield, a fault tolerant switch lattice for the building block is designed.

1. Block Requirements

a) Block Yield

With the column exclusion strategy, every faulty block causes the loss of an entire column of blocks. There is a multiplier effect associated with faulty blocks. (Once again, a faulty block does not have to be completely dysfunctional, but it is a block which due to faults does not contain an embedded virtual lattice.) As a result, very few bad blocks can be allowed. Otherwise a large percentage of the wafer will be unused.

What is the required block yield? To estimate this, assume a wafer contains an 8×8 grid of blocks. (In Chapter 5 on the Wafer Scale CHiP Processor, it will be shown that this is a reasonable and somewhat conservative grid size.) For any given block yield, p, we can compute the probability distribution of the number of faulty blocks in the 64 block grid. Since defects on the wafer are randomly distributed, the probability of the individual blocks being good are independent events. The status of a block is either functional or faulty, so the probability distribution of good and bad blocks is a binomial random variable. Pr(F = i), the probability of exactly i of the 64 blocks being faulty, is

$$\Pr(F = i) = \binom{64}{i} (1-p)^i \, p^{64-i}$$

To estimate the number of blocks left after column exclusion, we assume that i faulty blocks eliminate i columns (or rows) from the grid. (Note that it is certainly possible for two or more defective blocks to fall in the same column. This results in only one column not two being eliminated. This more detailed analysis of column exclusion is found in Chapter 5. It differs from the following estimate by only about 5%.)

Table 4.1.1 shows the results of this analysis for different block yields. Because of the multiplying effect of faulty blocks, the grid size obtainable is highly sensitive to the block yield. Even if 95% of the blocks are good, this still results in the loss of a large portion of the wafer; over 40% of the wafers use less than two thirds of the grid. Even with 97% block yield, 25% of the wafers will use only about two thirds of the blocks, and only 14% of the time will the all blocks be functional. This shows that even a small percentage of defective blocks causes a large reduction in the size of the grid after column exclusion.

Block yields of 98% and 99% show significant improvement. They are compared in more detail in Table 4.1.2. With 99% yield, over half of the wafers are fully functional, and with 98% yield over one quarter have no bad blocks. The expected number of usable blocks is 54.0 for 98% yield and 59.1 for 99%. This relatively small difference results form the fact that with 99%

Table 4.1.1 - Effect of Block Yield on Grid Size
(Worst Case)

number of faulty blocks	resulting grid size	block yield 0.95	0.97	0.98	0.99
0	8 × 8	.0375	.142	.274	.526
1	8 × 7	.126	.287	.358	.340
2	7 × 7	.210	.275	.230	.108
3	7 × 6	.228	.175	.0972	.0226
4	6 × 6	.183	.0828	.0303	.0040

Table 4.1.2 - Comparison of 98% and 99% Block Yield

block yield = 0.98		block yield = 0.99			
prob	cumulative prob	prob	cumulative prob	resulting grid size	% of grid used
.274	.274	.526	.526	8 × 8	100%
.358	.633	.340	.865	8 × 7	87.5%
.230	.863	.108	.973	7 × 7	76.6%
.0972	.961	.0226	.996	7 × 6	65.6%
.0303	.991	.0040	1.000	6 × 6	56.3%

yield, few lattices are smaller than 7 × 7. As a result, the 98% case receives a much larger contribution to its expected value from the 7 × 7 and 7 × 6 grids. This makes up for its smaller contributions from the 8 × 8 and 8 × 7 grids.

Although the expected number of usable blocks is similar, there are twice as many completely functional wafers with 99% yield than with 98% yield. This is important since a fully functional wafer enjoys a substantial performance gain over wafers with one or more faults. Excluding a column introduces a performance penalty. When a column is excluded, the two adjoining columns must be connected together. The length of wire (and the number of intervening switching levels) to implement this connection is substantially longer than if the columns are adjacent. The connecting wires must traverse at least the entire width of a column whereas adjacent columns are separated by very short distances. This longer wire length increases the signal propagation time. Inter-PE communication speed is decreased, and system speed goes down. Consequently, it is desirable to have wafers with no faulty blocks even though redundancy must be increased to achieve the higher block yield. To achieve this

- 99.0% or better yield is required for the building block.

b) Wire Around Capability

When a column is excluded, the two adjacent columns must be connected together. To accomplish this, the switches and datapaths in the unused blocks are used to make the required connections. The PEs in the blocks are not used but the functional switches provide the wiring bandwidth to connect together the two adjacent columns. Thus the "wire around" requirement becomes a "wire through" capability via the CHiP switch lattice. Figure 4.1.1 depicts an example of wire through.

If each block emulates an N × N virtual lattice with corridor width w, wN + 1 connections must be made. Each one of these requires a path from one side of the block to the opposite side. Since either rows or columns may be eliminated, any block must be able to provide the needed paths between both its East and West sides and between its North and South sides. Figure 4.1.1 shows the five connections that must be made for a 2 × 2 single corridor lattice.

Switches and datapaths are subject to failure just as processing elements are. Switch redundancy within each block is required so that wire through can be implemented despite the presence of faulty switches. Determining the degree of redundancy required is one of the building block design decisions that will be considered later.

2. Processing Element Design

The goal of the research in CHiP architectures is to investigate problems in parallel computation such as: parallel programming, inter processor communication, testing of concurrent systems, etc. CHiP machines are an assembly of many conventional microprocessors. Each is a von Neumann machine sequentially executing instructions dictated by the contents of its program counter. The substantial body of knowledge and design experience with such machines is built upon by using conventional processors as fundamental units in a parallel system. As a result,

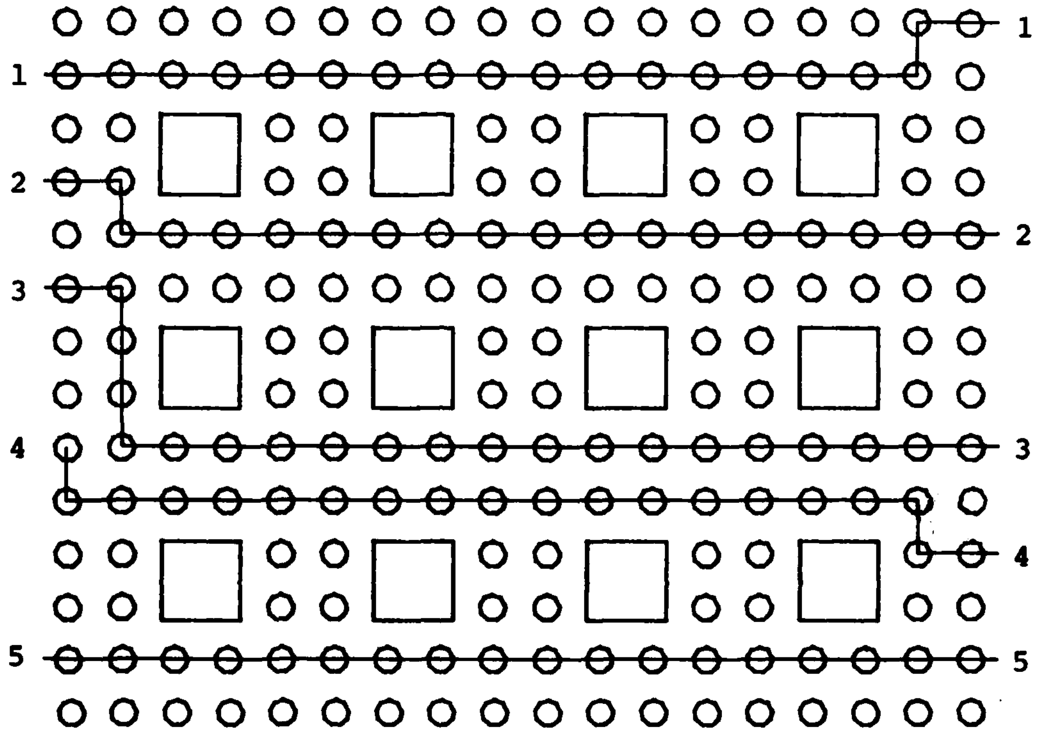


Figure 4.1.1 - Example of Wire Through in a Building Block

processing elements are largely treated as "black boxes." We are not concerned with details of the inner workings of the processing elements, nor do we want to design a processing element - this has been done many time by others.

However, knowledge of the area occupied by a processing element is essential to the quantitative analysis of the implementation of wafer scale machines. Fault tolerance is a necessity in a wafer scale system. It is achieved through redundancy, and the degree of redundancy required depends on the yield of the processing elements. Yield and area are closely linked.

Area estimation involves us in the design of processing elements. It is impossible to know the exact area of a processing element without specifying all the design details of the machine. Choice of word length, instruction set, control structure, etc. have a profound effect on the area occupied by the machine. However, the design of a processing element is a complex and lengthy task. Since the design of conventional and simple processors is a well explored topic, we will not to repeat it. To circumvent this, our goal is to estimate the area without producing a complete and detailed design of a specific processor. This will be done in four steps:

1) Analyze the functional requirements of the processing component of a CHiP processor. The intended applications of the machine determine the capabilities the machine must provide.

2) Determine the major architectural features. Very high level design decisions such as word length and memory size determine the gross characteristics of the processing element.

3) Sketch the layout of the processing element. A simple schematic floor plan showing the major elements of the implementation of the processing element such as control logic, memory, registers and ALU is proposed. Details of the implementation of the major blocks and their interconnection are not covered.

4) Determine the size of the primitive cells. Each of the subsections of the floor plan is composed of basic cells such as memory bits, a bit slice of the ALU, PLA term, etc. The dimensions of these primitive cells can be closely estimated from a previous design project by the author [Hedl81a] and from published reports on processor implementation [Fitz81].

Combining the floor plan and the dimensions of the individual cells, the area of the major blocks of the PE can be closely estimated. Adding to the size of the components an estimate of the wiring area required for their interconnection, the total PE area can be estimated.

a) Functional Requirements

The intended applications of CHiP processors determine the computational requirements of the individual processing elements. For example, the granularity of parallelism of the applications is a primary determinant of the processing element's required memory capacity. If a relatively large computation is preformed by each processing element, there must be substantial memory to hold the object code of the computation and store the intermediate results. Similarly, if there are only a small number of processing elements will be concurrently active, each must be fast (and therefore complex) in order for the entire assembly to have high throughput.

CHiP processors are capable of implementing a wide variety of applications: database operations [Hsia82], signal processing [Snyd82b], dataflow programs [Cuny82], and numerical applications [Gann81] are among the problems suitable for processing by the CHiP family of architectures. A major application of CHiP machines is the execution of systolic algorithms [Snyd82a]. Systolic algorithms implement the control structure of an algorithm primarily through the topology of the processing element array and the synchronization of the processors. As a result, different systolic algorithms require different interconnection patterns of processors. The switch lattice of CHiP machines provides the interconnection flexibility required for a processor array to reconfigure into a wide variety of different topologies. Additionally, many of the algorithms for the above applications area are systolic in nature. Systolic computation is fundamental to CHiP machines.

The basic characteristics of systolic algorithms are [Kung79, Kung82, Mead80]:

- simple and regular pattern in the flow of data and control signals
- highly pipelined computation
- only a small operation is performed at each computational site. This is consistent with the pipelined nature of the computation. Each stage in the pipeline performs only a small portion of the entire computation.

- the input data, intermediate results and output values are continuously and rhythmically passed from one computational site to another. This is the source of the term "systolic." There is a regular pumping of data through the processor in a manner analogous to the pumping of blood in a living organism. Data circulates rather than being stored in a central memory.

An example of a systolic algorithm is matrix multiplication performed on a hexagonal array of processors (example from [Mead80]; algorithm due to Kung, et. al.). The problem is that of multiplying two n × n matrices with bandwidth w (see Figure 4.2.1). The elements in the bands of the matrices A, B and C move through the network in three directions simultaneously. Each element of C is initialized to zero. Every processor performs an inner product step multiplying the incoming values of A and B and adding the result to the incoming C value. A careful study of the flow of data and its timing will convince the reader that each c_{ij} is able to accumulate all its terms before it leaves the processor through the upper boundary (see [Mead80] for a more complete discussion). The following observations about the algorithm influence the design of processing elements to execute the algorithm:

- Each processing element performs one addition and one multiplication (and, of course, any read / write operations required to transfer the operands). Thus the program of each processing element is very short and simple.

- Only three data values are stored in a processing element at any one time. The entire array collectively can hold a large amount of data, but

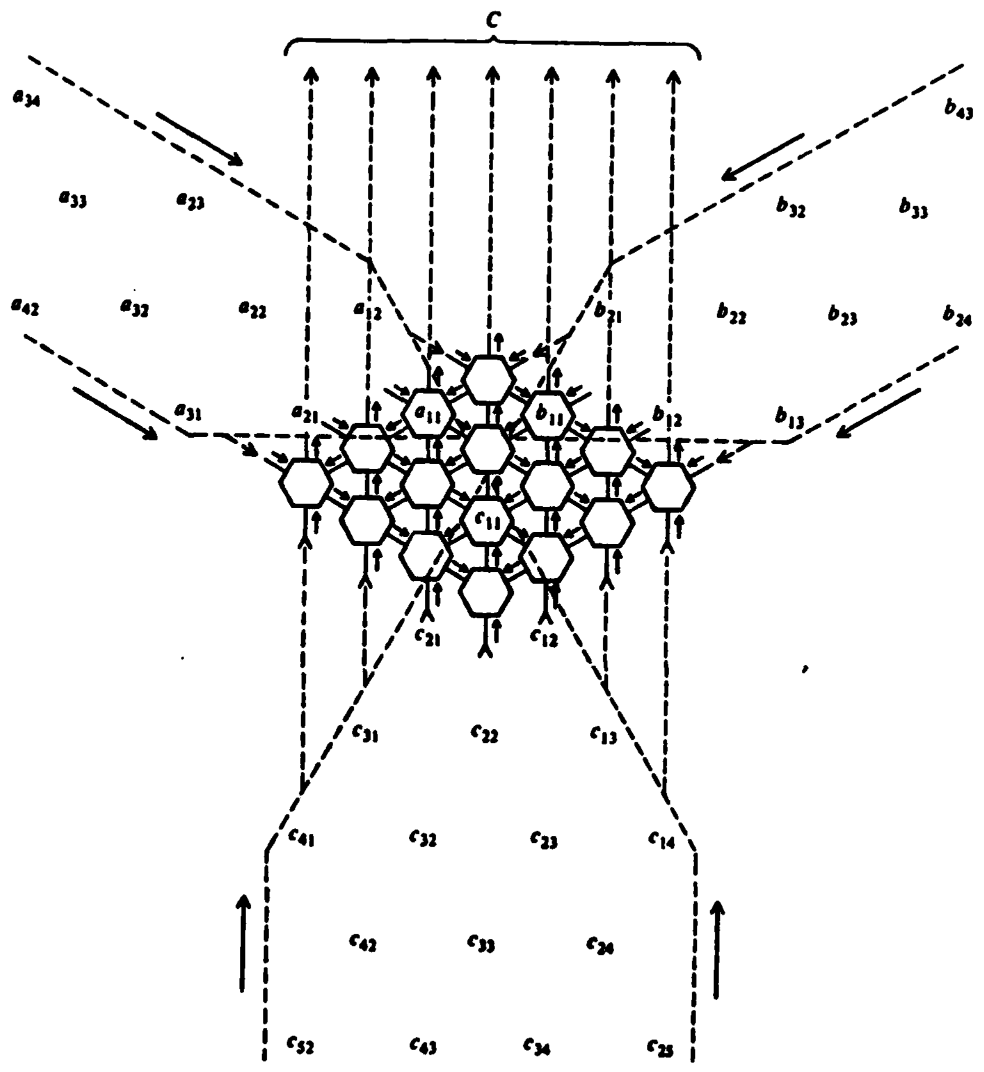


Figure 4.2.1 - Systolic Algorithm for Band Matrix Multiplication (from [Mead80])

each individual processing element stores only a few values. This exemplifies the principle of processing power through the collective action of many simple components rather than a few complex devices.

- High throughput is achieved through parallelism. A large number of processing elements are concurrently active. It is not necessary for each of the individual units to be fast in order for the entire assembly to achieve a high processing rate. Once again, strength through numbers.

- The computation is highly pipelined. As a single value of C passes through the array, it accumulates more and more product terms. By the time it reaches the upper boundary of the processor, the correct value has been accumulated. Pipelining especially in combination with large scale parallelism favors simple computational elements with modest speed.

A large body of systolic algorithms for a wide variety of problems has been developed in recent years. Algorithms exist for pattern matching in a string, LU decomposition, transitive closure, minimum spanning tree, dynamic programming, etc. (see [Kung82] for a comprehensive bibliography). All systolic algorithms exhibit the above general characteristics.

b) Processor Characteristics

What are the implications of the characteristics of systolic algorithms for the design of the processing element? The following basic architectural features are proposed as being well suited to the implementation of systolic algorithms:

1) Simple arithmetic oriented instruction set. The computational sites in systolic algorithms in general do not perform long, complex sequences of operations. Furthermore, many of the control operations of the algorithms are implicit in the topology and synchronization of the processing elements. This reduces the need for complex condition codes and branching instructions. Furthermore, a simple streamlined instruction set is consistent with an increasingly popular trend towards simplified machine architectures. A very small number of different instructions account for a very high percentage of instructions executed. These commonly used instructions typically perform simple operations. This phenomena has been observed for many different machines ranging from microprocessors to mainframes. Additionally, it has been found to hold for the object code produced for a large number of different high level languages [Peut77a, Peut77b, Knut70]. The philosophy of simplified machine architecture is to directly implement in the PE hardware only the most commonly used instructions. More complex operations are performed by sequences of the simple instructions. This philosophy is exemplified by the RISC [Patt81], MIPS [Henn81] and 801 [Radi82] architectural projects.

2) 8-bit ALU. An 8-bit word is both the ALU width and the size of words transferred between individual PEs and between PE and external memory. As previously noted, the parallel and pipelined nature of systolic computation deemphasizes the speed of the individual computing elements. Computations on longer operands are performed

one byte at a time - digit pipelined arithmetic [Owen81]. This further increases the pipelining of the machine. Furthermore, implementation considerations favor short word size. The restricted number of connections of the parallel processor to its external memory, and the limitations on memory bandwidth place a restrictive upper bound on the amount of data that can be practically transferred to or from the processing array in unit time. The rate at which the processor array requires operands must be matched to the limited memory bandwidth. A small word size decreases the number of memory bits transferred for each operand. Additionally, the area occupied by wiring between processing elements is dependent on the word size. Switch area is proportional to the square of the word size. A small word size decreases wiring overhead.

3) Five internal registers. There is one register for each port and an accumulator to hold temporary results. The port registers serve to buffer PE to PE communications.

4) 64 bytes of random access memory. This is the main memory of each processing element, and it holds both the PE's program and temporary data storage which can not be contained in the registers. The simple instruction set and the digit pipelined nature of the arithmetic computation increase the amount of program memory required. Some high level languages operations can not be performed by a single machine instruction but require a sequence of simple instructions. Plus digit pipelined arithmetic implements a single arithmetic operation in a sequence of single digit operations. However,

the main memory can hold 32 16-bit instructions which should be more than sufficient for systolic algorithms.

In many regards, the PE is similar to an 8-bit microprocessor such as the 8080. Both have simple instruction sets, and 8-bit ALU, a limited register file and byte wide data transfers. However, a CHiP processing element has important differences from a general purpose microprocessor. The environment of the PE is much more constrained. The limitations imposed by tailoring the PE for systolic algorithms provide a more restricted computational environment than that in which general purpose devices operate. These restrictions allow the following simplifications in the design of a PE:

- There is no need to provide a flexible and complex interrupt mechanism. The environment surrounding a PE is simple and fixed. A processing element communicates only with neighboring PEs or external memory. On the other hand, the general purpose microprocessor must be capable of interfacing to a wide variety of different devices from laboratory instruments, to terminals, to other input / output devices. Furthermore, it must be able to communicate with several of these devices simultaneously and perhaps with differing priorities. One of the microprocessor's strengths is generality. As a result, microprocessors commonly have a flexible, prioritized interrupt mechanism. This greatly increases the usability of the device but also increases its complexity. The constrained and limited forms of communication required of a CHiP processing element allow it greatly simplified communication and interrupt handling.

- Microprocessors generally provide a rich assortment of addressing modes to allow flexibility and convenience in fetching operands from the central memory. But with systolic computation, operands are continually being passed from PE to PE rather than residing in a central memory. The need for sophisticated memory access techniques is greatly reduced.
- Processing elements have a simple instruction set. As noted previously, there is reduced need for complex condition code setting and branching instructions.
- With the exception of PEs on the lattice edge, no signals are transferred off-chip. This eliminates bonding pads and pad drivers from the majority of PEs reducing their area.

In summary, CHiP processing elements due to their constrained environment and simpler computational requirements can be considerably simpler than conventional microprocessors. Simplicity leads to reduced area and greater reliability. Additicnally, a simple machine has fewer gates in the critical path of an instruction execution. Simplicity increases speed.

c) Layout and Area Estimation

Experience with the design of a simple prototype processing element [Hedl81a] suggests the PE layout shown in Figure 4.2.2. (Note that this rough floor plan is intended to be schematic in nature. The exact sizes of the components and their arrangement are approximately but not precisely reproduced. The point is to "rough out" the design of a processing element but not to provide the detailed design.) The register file contains the

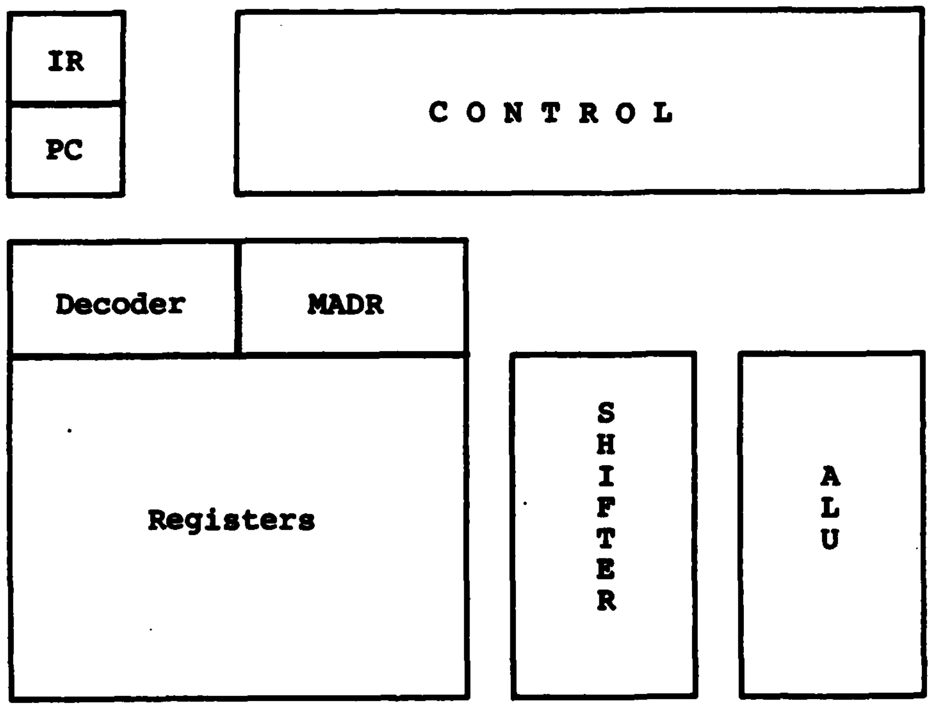


Figure 4.2.2 - Processing Element Layout - a Schematic Floor Plan

Table 4.2.1 - Area Estimation for a Processing Element

	Area(Kλ)
Main Memory (64 bytes)	
Memory Array	478
Address Decoder	187
	665
Datapath	
Shifter	55.6
ALU	110
Registers	67.2
	233
Control Logic	
PLAs	230
Latches	420
Wire Routing	470
Scan in / Scan out	40
	1160
Total for Components	
Main Memory	665
Datapath	233
Control Logic	1160
	2058
Wire Routing(20%)	412
	2470
Misc. Expansion (20%)	530
Total PE Area	3000

accumulator, the four port buffers and the 64 bytes of program and data memory. Both instructions and operands are fetched from the register file. One of the operands can be passed through the shifter before entering the ALU. The output of the ALU is stored back into the register file. The control logic section is a set of PLAs which decode the contents of the instruction register and time the sequence of data transfers to implement the current instruction. The distinguished registers of the machine include the instruction register (IR), program counter (PC), memory address register (MADR) and the accumulator (AC).

To estimate the sizes of the components of the layout, we draw on the experience of the RISC design team [Fitz81]. Both the CHiP processing element and the data path of the RISC machine share similar design objectives. Both machines have simple instruction sets and datapaths of reduced complexity, and both attempt to support high level language programming with minimum processor complexity. Additionally, the RISC team reported very detailed data on the layout complexity and size of their design. In their design, they spent considerable time and effort in the layout of compact and efficient components such as memory cells, ALU slices, etc. This has proved invaluable in making tighter and more realistic estimates of the area of the CHiP processing element.

Table 4.2.1 shows the area estimates of the major functional components of the processing element. All estimates were derived from the RISC Blue design group. Their layout was restricted to using only horizontal and vertical lines, a Manhattan geometry. This restriction was forced due to the computational complexity of the automatic circuit extraction and design rule checking programs. Additionally, Mead and Conway design rules were employed. A more realistic, industrial design environment would use a richer and much more complex set of design rules which are fine tuned to a particular fabrication process. Process specific rules have tighter spacings and smaller wire widths than the Mead and Conway "generic" rules. Designing with fewer restrictions and tighter design rules, better results both in area and performance are certainly obtainable. The following estimate may be regarded as an upper bound.

The area estimates in Table 4.2.1 were derived by scaling the functional block area reported by the RISC blue design team. For example, the RISC register array consists of 138 32-bit words and occupies 4.12 $M\lambda^2$. Each of the static RAM cells is a standard six transistor design with two independent data busses allowing two port access to the register file. Conceptually, this allows the accumulator and port registers to occupy the same memory array as the program / data memory. This reduces processor complexity. The RISC word size is longer and the number of registers is larger, so the area occupied by the memory of a CHiP processing element is estimated by scaling down the area figures for RISC. Direct scaling of the memory area reported by the RISC project shows that the 64 byte memory array of the CHiP processing element will occupy area

$$\frac{64}{138}\frac{8}{32}4.12\ (M\lambda^2) = 478\ (K\lambda^2)$$

Similarly, the ALU area scales linearly down from the 32-bit wide RISC datapath that occupies 0.44 $M\lambda^2$ to

$$\frac{8}{32} 0.44 \ (M\lambda^2) = 110 \ (K\lambda^2)$$

The critical component of the RISC design is the memory. It occupies most of the are of their design. As a result, considerable effort was spent optimizing the memory cell layout and the memory fetch/store timing. The memory area estimate can be considered to be quite near optimal. But the pitch of the memory cell determined the height of the ALU. The ALU area was not independently optimized, but rather its layout was dictated by the requirement to mesh with a previously designed memory unit. This is not necessarily optimal for the CHiP processing element. In short, there may be room for improvement in the ALU estimate.

Not all components of the layout scale linearly. Decoder size is proportional to the square of the number of inputs. The RISC memory contains 138 words, and its memory decoder occupies area 0.87 $M\lambda^2$. From this the size of the address decoder for a 64 byte memory is roughly

$$\left(\frac{64}{138}\right)^2 0.87 \ (M\lambda^2) = 187 \ (K\lambda^2)$$

The shifter area also scales quadratically (from 0.89 $M\lambda^2$ for a 32-bit shifter of RISC) to

$$\left(\frac{8}{32}\right)^2 0.89 \ (M\lambda^2) = 56 \ (K\lambda^2)$$

Note that all components contain an area component which is independent of the number of inputs. A more accurate scaling model is Area = An + B where n is the number of inputs. The above analysis is an approximation with B = 0.

In addition to the above components, the processing element architecture includes a number of one byte registers: four port registers, the accumulator, program counter, memory address register plus the two byte long instruction register. A single byte register is estimated to occupy area

$$\frac{1}{138}\,\frac{8}{32}\,4.12\ (M\lambda^2) = 7464\ (\lambda^2)$$

so the nine bytes required for the auxiliary register occupy 67.2 K λ^2.

Memory occupies a significant, 24.4%, of the PE area. To double check the memory estimates, we calculate the estimated size of a single bit of memory. Direct scaling estimates its area to be

$$\frac{1}{138}\,\frac{1}{32}\,4.12\ (M\lambda^2) = 933\ (\lambda^2)$$

With the reported vertical pitch of a register bit being 44 λ, this results in each memory bit occupying a 44 λ × 21 λ region. Since this is quite reasonable for a six transistor, dual bus memory cell, our estimates are accurate.

The instruction sets of both the CHiP processing element and the RISC machine are similar. Consequently, the control logic for the two machines will be of similar complexity. The control logic area for the CHiP PE is taken to be identical to the RISC values. This figure includes PLAs, latches to buffer the control signals, wire routing and scan in / scan out circuitry to enhance testability.

The total area of the above components is 2.058 Mλ^2. 20% additional area is added for additional wire routing between the major functional blocks. (This is the same percentage as reported by the RISC group.) Since layouts always occupy more area than expected,[1] an additional 26% area is added to bring the total area estimate to a round 3.00 Mλ^2.

From this estimate. a PE occupies a square of side 1732 λ. Bringing this estimate up to the nearest round number,

- each CHiP processing element is estimated to occupy a 1750 $\lambda \times$ 1750 λ region of silicon.

This final rounding results is an additional $(1750)^2 - (1732)^2 = 62.5$ kλ^2 area for each PE. Our area estimate is conservative. Above and beyond the estimated size for all components and wire routing between them (2.470 M λ^2), an additional $0.530 + 0.0625 = 0.5925$ Mλ^2 has been added to the estimate. This is an additional 24% for miscellaneous expansion. The estimate contains considerable "free area" for unanticipated uses.

3. Datapath Design

Datapaths are the busses connecting switches and processing elements. In addition to data, these signals also include control signals for the PEs and switches. Each of the individual bus wires is independent of the others.

(Note that the term "datapath" is used ambiguously. In the context of processing element design, the datapath is the portion of the machine that transforms and modifies data - the shifter and ALU. Within the context of lattice design, where PEs are treated as black boxes, datapaths are simply

[1] a basic law of nature

busses transmitting data without alteration. The intended meaning of the term will be clear from the context of its usage.)

The datapath is quite small in comparison to the processing elements and switches. A PE occupies a square 1750 λ on a side while in the following section it will be shown that switches are approximately 250 λ on a side. To estimate the datapath width, assume there are ten signals per datapath. This is sufficient for one byte of data and two control signals - one for the processing elements and one for switches. The distance between PEs is much longer than the distances encountered when routing data within a single PE. To reduce signal transmission time, datapaths are implemented in the metal layer since metal has much lower resistance and capacitance than the polysilicon or diffusion layers. With Mead and Conway design rules, each metal wire is 3 λ wide, and the separation between wires is also 3 λ. Therefore a ten wire datapath has a minimum width of approximately 60 λ. This is one quarter the width of a switch and only 3.5% the size of a processing element (Figure 4.3.1).

In addition to being small, the datapath width can be increased without increasing the lattice area. Widening the datapaths in Figure 4.3.1 does not increase the separation of switches and processing elements and so has no effect on the size of the lattice. Note that this is dependent on the layout details and shapes of the PEs and switces.

As a result, datapaths can be designed with relaxed design rules without increasing lattice area. By increasing the width of wires, the probability of a break in a wire is reduced. By increasing the separation between wires, there is less chance that two adjacent wires will short out. Relaxed design

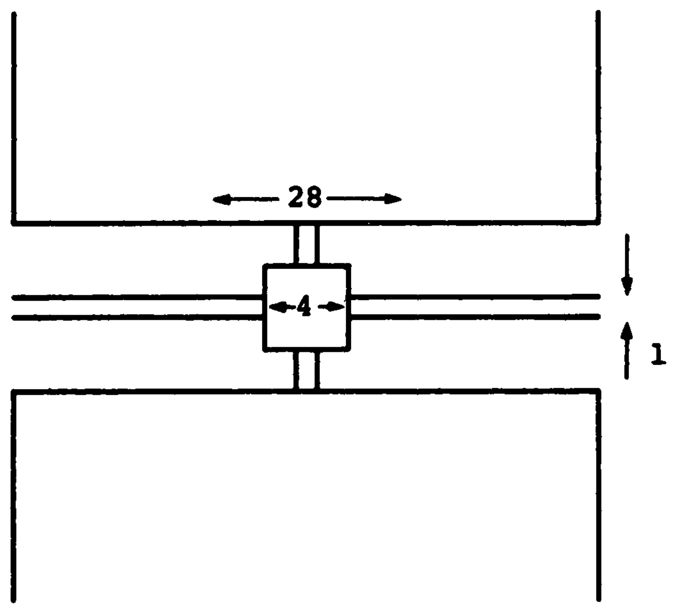


Figure 4.3.1 - Approximate Relative Size of a PE, Switch and Datapath

rules decrease the circuit's sensitivity to defects [Rung81]. The same number of defects may occur but the probability of a defect causing the circuitry to malfunction is reduced.

The relationship between the design rule spacing and yield for a given circuit is process specific. The amount of yield increase for a given increase in design rule spacing can not be predicted without also specifying the fabrication line on which the circuit will be manufactured. However, the large disparity between switch and datapath size gives great flexibility in the design rules for the datapath. The datapath width can be increased by a factor of four without effecting lattice area. This allows wire widths and spacings to be up to four times as large as allowed by the minimum resulting in large yield increases.

From the combination of datapaths being small, simple and designed with flexible design rules, we

- assume there are no fatal defects in datapaths.

This is of course an approximation, but with very high datapath yield, it is a very close approximation.

Note that an increase in design rule spacing of the datapath has no effect on the machine's performance. The signal propagation time is unaffected by the width of datapath wires. As the width of a wire increases, its capacitance per unit length increases proportionately. However, resistance decreases linearly with width. Since the signal propagation time is proportional to the product of the wire's resistance and capacitance, the signal delay is unchanged by increasing the wire width.

4. Switch Design

a) Switch Layout

A sample layout of a switch is shown in Figure 4.4.1. The switch displayed there is one of the simplest possible - degree four, no crossover capability and only one configuration setting. Extensions to a more complex switch are straightforward.

The switch architecture is organized around its bus rail - concentric squares of independent bus wires. There is one wire in the bus rail for each wire of the datapath. At each of the compass point directions, NSEW, the bus rail is connected to the datapath. This connection is controlled by the configuration setting. The four bits of the setting determine which subset of the four datapaths are connected to the bus rail. If bits N and E of the setting are "on" (with S and W "off"), these two datapaths are connected together via the bus rail while the S and W datapaths are disconnected from the bus rail. The configuration setting controls datapath access via four sets of pass transistors. Each of the groups of pass transistors is driven by one bit of the configuration settings as indicated by the labels on the control lines in Figure 4.4.1.

b) Switch Yield

A simple switch with degree 8 and crossover capability occupies an area of approximately 250 λ × 250 λ. To estimate the yield of an individual switch, recall that a normalized unit area contains a 2 × 2 white lattice and

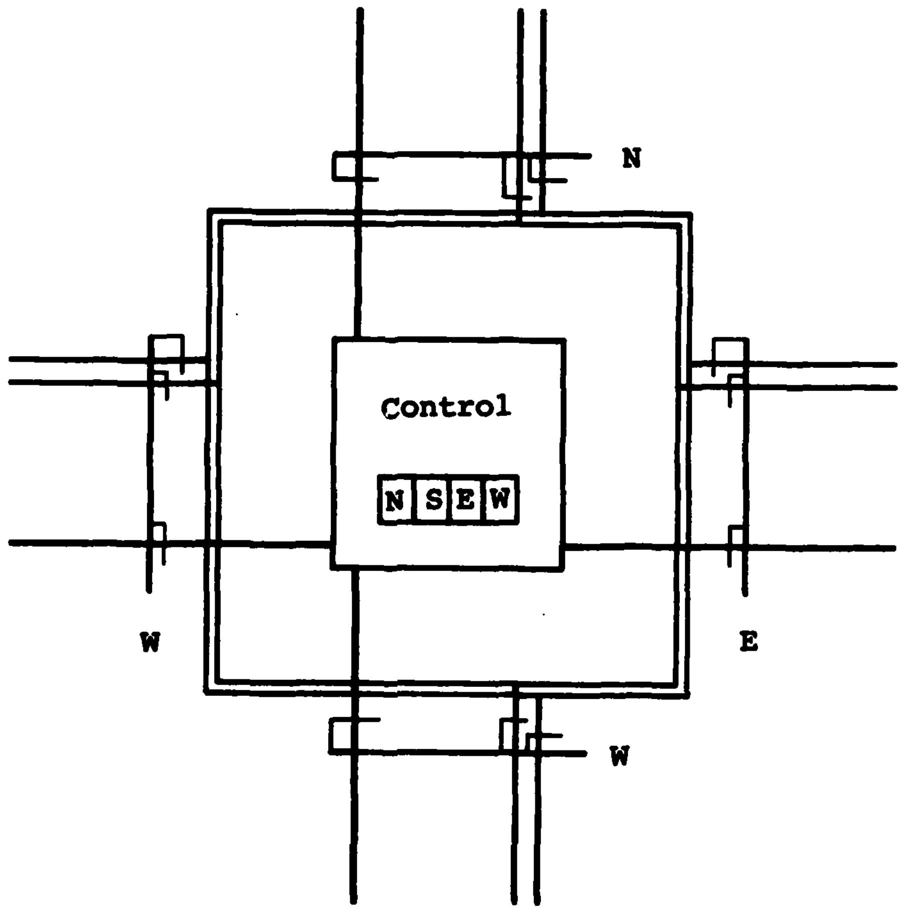


Figure 4.4.1 - Switch Layout - An Approximate Floor Plan

has 20% yield (see Chapter 2). Since we have assumed that no fatal defects occur in datapaths a unit area consists of four PEs and 21 switches. With PEs occupying a 1750 $\lambda \times$ 1750 λ region, the lattice area sensitive to defects is

$$4\,(1750)^2 \;+\; 21\,(250)^2 \;(\lambda^2) \;=\; 13.56 \;(M\lambda^2) \;=\; 1 \;\;\text{(unit area)}$$

The area of a single switch is

$$A_S = \frac{(250)^2}{13.56 \times 10^6} = 4.61 \times 10^{-3} \;\;\text{(unit area)}$$

Substituting this into the yield model

$$Y_S = \frac{1}{(1 + A_S\, s_0)^4} = 0.991$$

This indicates that switches will have over 99% yield.

The yield equation results from the mathematical modeling of the manufacturing of *typical* integrated circuits. Yield commonly varies in the 2% to 50% range. Extrapolating this model to exceptionally high yields may be unreliable. The 99% estimate may be either low or high. Although the specific yield figure may be questionable, the general conclusion that can be drawn is that switches have a very high yield. There is also another factor that supports high yield.

Switches are quite small compared to processing elements. As a result, a proportionately large increase in switch area results in only a small increase in total lattice area. Furthermore, some switch expansion results in absolutely no increase in lattice area. In Figure 4.3.1, the switch can be expanded horizontally without increasing the PE to PE spacing.

Consequently, relaxed design rules can be used for switch design. As with datapaths, switch yield can be increased with little or no impact on area. In short, through their small size and use of relaxed design rules

- very high switch yield can be assured.

To roughly estimate the size of a 2 × 2 white lattice, note that the lattice has two rows of PEs and three of switches. The total edge length is at least $2(1750) + 3(250) = 4250\ \lambda$. Allowing for spacing between components, datapath routing, power lines, etc., we conservatively estimate that a 2 × 2 white lattice occupies a square of edge 4750 λ. With 1 technology, the edge length is 4.75mm.

5. Lattice Design

So far in this chapter, the requirements for building blocks have been specified, and the design of the individual processing elements, datapaths and switches has been discussed. This section considers the integration of these individual components into a building block meeting the requirements of 99% yield and reliable wire through capability.

The first design decision to be made is the dimensions of the virtual lattice which is mapped into the building block. After this, the characteristics of the building block which hosts the virtual lattice must be decided upon. This involves the degree of PE redundancy required to achieve high block yield. Additionally, a switch lattice must be chosen that provides sufficient wiring flexibility despite switch faults to implement both the mapping of the virtual lattice into the block and wire through capability. These considerations are discussed in detail below.

The size of the virtual lattice determines in part the size of the building block. A larger virtual lattice with more PEs necessitates a larger building block.

Large building blocks entail several disadvantages. First, after the mapping of the virtual lattice into the block. The maximum path length between PEs is bounded by the size of the block. Larger blocks permit longer paths. System speed is reduced by long paths. Hence, there is a strong preference for small blocks that can be mapped using only short wires.

Secondly, the complexity of determining the mapping of the virtual lattice into the block increases with block size. With a larger block more different mappings are possible. Since the mapping problem is solved by basically brute force methods, increases in block size may substantially increase the time required to determine the mapping. As a result of these considerations, a small virtual lattice is chosen (Figure 4.5.2).

- a 2×2 virtual lattice will be mapped into the building blocks

A building block must be chosen that effectively hosts the virtual lattice. What are the requirements for a virtual lattice to be mapped into a building block? Each component in the virtual lattice must have a counterpart in the block. Therefore, at a minimum,

1) a block must contain at least as many functional PEs as the virtual lattice

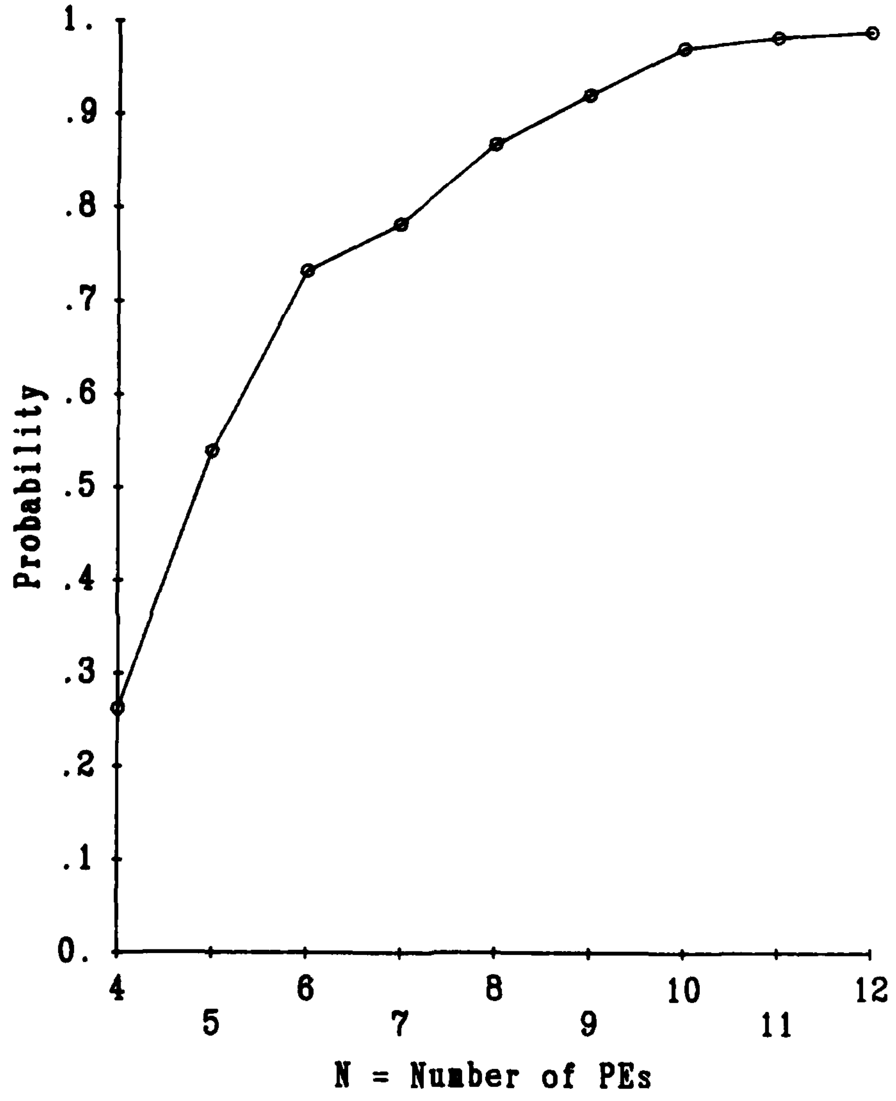


Figure 4.5.1 - Recovery Curve for Four PEs in N PEs

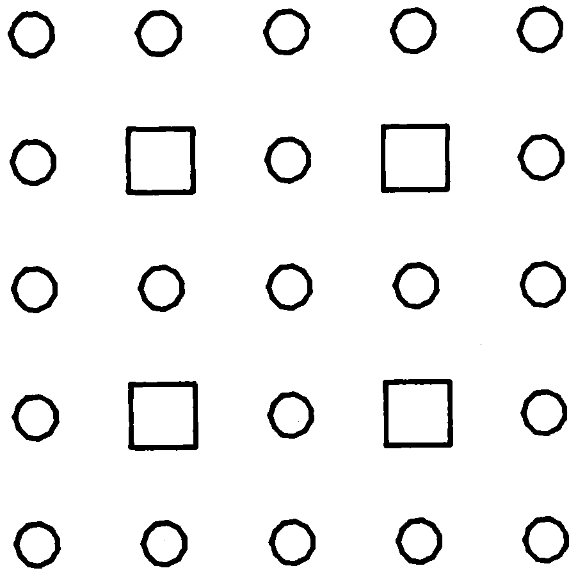


Figure 4.5.2 - Virtual Lattice to be Mapped
Into a Building Block

and 2) as many functional switches as the virtual lattice.

In addition to the block switche' which are images of switches in the virtual lattice, there must be enough functional switches left in the block to act as the connecting switches. These implement the datapaths of the virtual lattice. They serve as the "glue" to wire together the components of the virtual lattice. In short,

3) the datapaths of the virtual lattice must be mappable into the building bock.

The virtual lattice must be recoverable from the block with at least 99% probability. For a successful mapping, each of the three requirements must be met by a block. If a block fails to meet any one of the requirements, it will be impossible to map the virtual lattice into the block.

By far the most difficult of these three requirements is that the block has the requisite number of PEs. There are likely to be very few defective switches or datapaths, and the yield of PEs is much lower than switches or datapaths.

In the subsequent sections, a switch lattice that is highly robust will be proposed for building blocks. Switches are small so the addition of redundant switches causes little increase in the lattice area. The area of a switch can be increased by a large *percentage* while increasing the lattice by only a small fraction. Furthermore, much of this increase is in the portion of the lattice occupied by the datapath. This part of the lattice is highly insensitive to defects. Increasing its area causes very little increase in the number of fatal defects; lattice yield is almost unaffected. As a result, it is inexpensive (in terms of area) to provide essentially 100% reliability through redundancy in the switch lattice. Consequently, if a block contains enough PEs, the mapping of a virtual lattice will be almost assured. PEs are the weak link. We consider them next and then return to the switch lattice design.

a) Processing Elements

We must determine the number of PEs, N, per block such that four good PEs can be found out of the set of N PEs with 99% probability. This is an instance of the recovery problem discussed in Chapter 2. Drawing on the results of recovery analysis, Figure 4.5.1 shows R (probability of recovery of four PEs) vs. the total number of PEs. A total of 12 PEs gives the required 99% recovery so

- each building block contains a 4 PE × 3 PE CHiP lattice.

b) Switches

From section 4.4, switch yield, Y_S, is estimated from the yield model to be 99.1%. This yield is achieved through the combination of small switch area, simplicity and use of relaxed design rules. Throughout this section, calculations will be made for the purposes of comparison based on both 99% and 97% yield for. This is a more conservative approach than flatly assuming 99% yield, and it will indicate the sensitivity of the design decisions to changes in switch yield.

As noted in previous sections, switches have high yield. But no matter how high the yield, the random nature of defects means that functionality can not be guaranteed; some switches will always be faulty. Consequently, each switch must have at least one other in the lattice that can take its place. To provide adequate switch redundancy,

- the corridor width of the building block is two (Figure 4.5.2); twice that of the virtual lattice.

This provides 100% switch redundancy. The building block has twice as many switches as necessary.

Note that this redundancy has low cost. Switches are quite small in comparison to PEs. Adding extra switches causes only a small increase in overall lattice area. In Figure 4.3.1, increasing the width of the switch corridor between the PEs from one to two increases the separation of the PEs. This increases the area occupied by the lattice by (no more than) 4×28 units for every row and column of switches. This increase the lattice area approximately 14%. Most of this additional area is occupied not by switches but by the datapaths which are highly insensitive to the presence of defects. The portion of the lattice sensitive to defects (PEs and switches) is called its ***active area***. This increases by only 2% $(= 4^2 / (2(4^2) + 28^2)\ )$. As a result, the yield of the lattice is effected very little by the increase in corridor width. Furthermore, depending on the details of the switch layout, it may be possible to pack the second switch into the inter PE area in such a way that it causes a smaller increase in the PE separation. In turn, lattice area would increase less. In summary, both overall lattice area and lattice yield change little as a result of increasing corridor width.

As noted in the previous section, PEs are the "weak link" in a building block. The key to a high block recovery rate is having the required number of functional PEs. For a PE to be good, all four of its ports must be functioning correctly. A port itself may be functional but it is unusable if the switches to which it is connected are faulty. In the virtual lattice (Figure 4.5.2), failure of any one of the four switches directly connected to a port renders the entire PE dysfunctional. A PE is not usable unless it can communicate with its surrounding environment from all four of its ports.

To safeguard against a switch failure rendering a PE unusable, the building block provides 100% switch redundancy at each port. Every port has two switches connected to it. Either one switch or the other can connect the port to the remainder of the switch lattice. Only one of the two switches must be functional. Clearly, it is still possible for both switches attached to a port to be faulty. At switch yield, Y_S, of 0.97, the probability of a PE having a port which is disconnected from the switch lattice due to a double switch fault is $4 \times (1 - Y_S)^2 = 0.36\%$. At $Y_S = 0.99$, this probability shrinks to 0.04%. We can not totally prevent switch faults from disabling PEs, but the probability is reduced to a very small value.

How many switches in a building block are likely to be faulty? The switch yield is the average number of faulty switches. But since defects are a random process, the exact number of faults per block will fluctuate from block to block. What is the maximum number of faulty switches that can "reasonably" be expected?

By the assumption of the random distribution of defects, the probabilities of the individual switches being defective are independent. Since switches can be in one of two states, good or bad, a binomial probability distribution applies to the collection of switches in a block. An n PE × m PE building block with double corridor width and two switches per port has a total of $(4n + 2)(4m + 2) - nm$ switches. The 4 × 3 building block has 240 switches. Let F be a random variable representing the number of faulty switches per block. With $Pr(F = f)$ representing the probability that a block has exactly f faulty switches, we have

$$Pr(F = f) = \binom{240}{f} (1 - Y_S)^f Y_S^{240-f}$$

The expected value of F is $240(1 - Y_S)$, and its standard deviation is

$$\sigma = \sqrt{240 Y_S (1 - Y_S)}$$

These values are shown in Table 4.5.1 for 99% and 97% average switch yield, Y_S. From this it can be seen that on the average there are only a small number of faulty switches per block. How does the actual number of faults vary from block to block? By Chebyshev's Theorem, at least $1 - (1/4)^2 = 15/16 = 94\%$ of the blocks are within ±2 standard deviations of the mean. At 97% switch yield, this means $94\% < Pr(7.20 - 2(2.64) < F < 7.20 + 2(2.64)) < Pr(F < 12.5)$, or at least 94% of the blocks have no more than 5% (= 12.5/240) of their switches faulty. For $Y_S = 0.99$, the same fraction of blocks has no more than 2% (= (2.40 + 2(1.54)) / 240 = 5.4/240) faulty switches.

Chebyshev's Theorem bounds the spread of F for any probability distribution of F. The exact distribution of F is shown in Table 4.5.2. Examining these more exact calculations, it can be seen that the spread of the defect distribution is somewhat less than predicted by Chebyshev's Theorem. The binomial distribution clusters more tightly about the mean value than the Chebyshev limits predict. With an average switch yield of 99%, almost all the

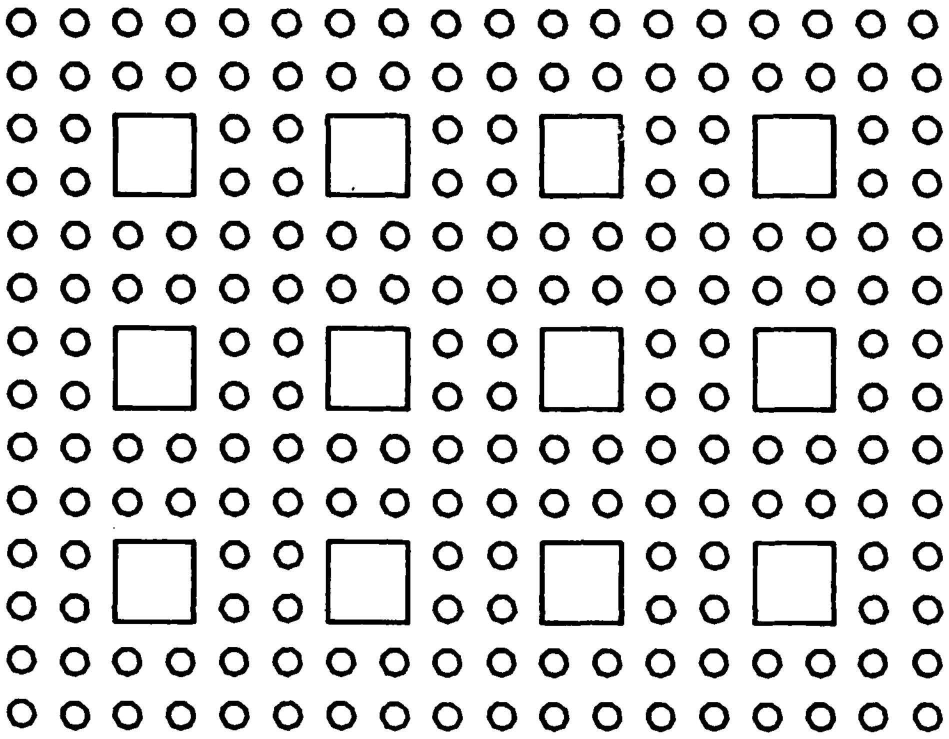


Figure 4.5.3 - Building Block for a Wafer Scale CHiP Processor

Table 4.5.1 - Effect of Switch Yield on the Number of Faulty Switches Per Block

	Switch Yield 0.99	Switch Yield 0.97
expected number of faulty switches per block (M)	2.40	7.20
standard deviation (σ)	1.54	2.64

Table 4.5.2 - Probability Density of Defective Switches

f = number of faults	Switch Yield = 0.99		Switch Yield = 0.97	
	Pr(F = f)	Pr(F ≤ f)	Pr(F = f)	Pr(F ≤ f)
0	.0896	.0896	.0037	.0037
1	.217	.307	.010	.014
2	.262	.569	.022	.035
3	.210	.779	.043	.078
4	.126	.905	.073	.15
5	.0600	.965	.11	.26
6	.0237	.988	.14	.39
7	.00801	.997	.15	.54
8	---	---	.14	.68
9	---	---	.12	.80
10	---	---	.087	.89
11	---	---	.054	.95
12	---	---	.029	.97
13	---	---	.014	.99

blocks (>99%) realize at least 97% (= (240-7)/240) switch yield. Although the actual switch yield can fluctuate in accordance with the binomial distribution, it almost never dips below 97%. Similarly, with Y_S = 97%, all but one percent of the blocks achieve 95% (= (240 - 13)/240) yield.

Derivation of Table 4.5.2

The distribution of F for $Y_S = 0.99$ was derived by directly applying the formula for the binomial distribution

$$\Pr(F = f) = \binom{240}{f} (1-Y_S)^f \, Y_S^{240-f}$$

For all but very small values of f, computing the binomial coefficient

$$\binom{240}{f}$$

is cumbersome and lengthy.

For $Y_S = 0.97$, the binomial distribution was approximated by a normal distribution [Ross76] with

$$\Pr(F = f) = \Pr(f-0.5 < F < f+0.5)$$

Let M be the mean value of F and σ its standard deviation. Converting to the unit normal distribution, Φ, we have

$$\Pr(F = f) = \Pr\left(\frac{f-0.5-M}{\sigma} < \frac{F-M}{\sigma} < \frac{f+0.5-M}{\sigma}\right)$$

$$= \Pr\left(\frac{F-M}{\sigma} < \frac{f+0.5-M}{\sigma}\right) - \Pr\left(\frac{F-M}{\sigma} < \frac{f-0.5-M}{\sigma}\right)$$

$$\approx \Phi\left(\frac{f+0.5-M}{\sigma}\right) - \Phi\left(\frac{f-0.5-M}{\sigma}\right)$$

where the values of Φ are obtained from a table of the normal distribution.

c) Mappability

The building block must contain the PEs and switches to serve as the images of the PEs and switches in the virtual lattice. Additionally, the datapaths of the virtual lattice must be implemented by the building block. These are mapped to either single datapaths in the block or a *path* of connected switches and datapaths; a single datapath of the virtual lattice may become a chain of switches in the block.

In addition to producing one single mapping, it is desirable to find a mapping that has a short maximum path length between components. As noted elsewhere, long paths reduce system performance.

The switch lattice of the building block can be chosen to help reduce path lengths. By increasing the wiring bandwidth of the switch lattice, shorter and more compact mappings can result. In particular, we propose for the switch lattice of building blocks:

a) switch degree eight. The switch degree is increased from four in the virtual lattice to eight in the building block. The addition of diagonal connections allows some routings to "cut the corner" to reduce path length. In Figure 4.5.4a, the diagonal datapath replaces one switch and two datapaths that would be required in a degree four lattice. Longer diagonal traversals reduce path lengths correspondingly.

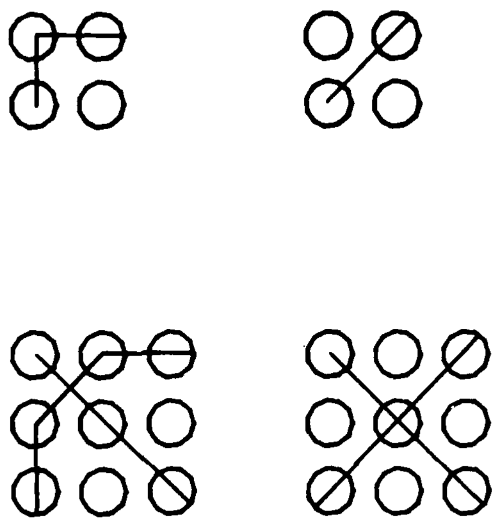


Figure 4.5.4 - Wire Saving Due to Switches With
Degree 8 and Crossover Capability

b) crossover capability. By allowing two independent paths to cross at a switch, paths can often follow the most direct route instead of detouring around crossover points. In Figure 4.5.4b, the crossover at the center switch saves one switch and one datapath.

Incorporating these characteristics in the switch lattice of the building block increases the efficiency of the resulting mappings. More compact mappings result with the corresponding increase in performance. We propose that

- building blocks have degree eight switches with a crossover capability of two.

Even with this increased wiring capability of the switch lattice, it is impossible to guarantee a mapping of the virtual lattice into the building block even when there are the required number of functional PEs. It is always possible that a mapping will be prevented by a particular pattern of faulty switches. For example, an entire row of faulty switches divides the block into two disconnected components. These particular patterns are extremely unlikely given the high switch yield and the large amount of wiring bandwidth provided by the switch lattice.

d) Wire Through

The requirement for wire through capability is that there exists five continuous paths from the left side of the block to the right side (Figure 4.1.1). The block is unused so all functional switches are available for implementing the paths. Orienting the block so the short side is vertical provides the least wiring bandwidth from left to right. This orientation is chosen for the following worst case analysis. Additionally, the paths are allowed to start and end at any switch on the edge. Note that this is a somewhat more liberal criterion than is actually required for wire through in which paths must maintain their relative positions. But adding restrictions to the format of the path simply decreases the probability that such paths exist. In effect, we derive an upper bound for the probability that the paths do not exist. We show this upper bound is acceptably small.

Model the problem as a graph with switches represented by nodes and the datapaths by edges. Since PEs do not participate in the wire through, they are not included in the graph. A faulty switch corresponds to removing that node from the graph. The problem is to find sets of nodes whose removal reduces the minimum edge bisection width of the graph to four or less. Call this *bisecting* the graph. Since the probability density of defective switches decreases rapidly as the number of defective switches increases, we first find the minimum set of nodes to bisect the graph. Bisections requiring more than the minimum number of switch faults will occur less frequently.

The narrowest portion of the graph is the eight columns from which a PE has been removed. The graph is divided by the missing PEs into four separate wiring channels each of which is is two switches wide. For the graph to be bisected at slice A, each of the four channels must have at least one faulty switch. The minimum bisection width of the graph is greater than four unless this condition is met. (Note that by using the crossover capability of the switches, the routing can be achieved with as few as three switches in any column - not five. The following is an upper bound.) The probability of a given channel having at least one fault is

$$\begin{aligned} \Pr(C) &= \binom{2}{1} Y_S (1 - Y_S) + \binom{2}{2} (1 - Y_S)^2 \\ &= 0.0296 \qquad Y_S = 0.97 \\ & \ 0.0199 \qquad Y_S = 0.99 \end{aligned}$$

The probability of all four of the channels having at least one fault is

$$\begin{aligned} (\Pr(C))^4 &= 7 \times 10^{-7} \quad Y_S = 0.97 \\ &= 2 \times 10^{-7} \quad Y_S = 0.99 \end{aligned}$$

With eight different slices, the probability of one of the slices bisecting the graph is

$$\begin{aligned} 8 \times (\Pr(C))^4 &= 6 \times 10^{-6} \quad Y_S = 0.97 \\ &= 1 \times 10^{-6} \quad Y_S = 0.99 \end{aligned}$$

To bisect the graph through one of the columns containing 14 switches (slice B, Figure 4.5.5), the probability is less than

$$10 \times \binom{14}{10} (0.97)^4 (0.03)^{10} = 5 \times 10^{-12}$$

Consequently, the probability of faulty switches causing the minimum bisection width of the graph to fall below five is negligible. As a result, we will assume that

- building blocks can always implement wire through capability.

CHAPTER 5

A WAFER SCALE CHiP PROCESSOR

In this section we consider the design of a wafer scale CHiP processor using the building block described in the previous chapter. The goal is to fabricate a large-scale parallel processor on a single wafer of silicon. This would allow the processor component of a parallel processing system to be constructed from a small number (perhaps one) of wafer scale components. Consideration is given to the problems of the layout of the blocks on the wafer, external connections, the actual number of processing elements per wafer, and the overall efficiency of this approach.

1. Wafer Layout

Each building block occupies three times the area of a 2×2 lattice. Since a 2×2 lattice occupies a square of side 4.75 mm, we approximate the size of a block by a square with edge $4.75\sqrt{3}$ or 8.23 mm. (The actual aspect ratio of the building blocks is highly dependent on the layout of processors and switches. Blocks may have one side slightly longer than the other. For simplicity we assume throughout this work that blocks are square. However, we avoid packing the wafer tightly with blocks. This leaves unused wafer area available in the proposed wafer scale machines to accommodate small adjustments in building block geometry.) The number of squares of edge length e that can be packed into a circle of diameter D is [Phis79]

$$\frac{\pi D^2}{4 e^2} - 1.77 \frac{D}{e} \tag{1.1}$$

The first term is th ratio of wafer area to chip area. The second term represents the number of chips that do not entirely fit on the wafer due to the curvature of the wafer edge. A 4" (101.6 mm) diameter wafer is the industry standard,[1] and it can hold a maximum of 98 of the 4 × 3 building blocks.

However, it is not desirable to pack as many blocks as possible on the wafer. Obviously, room must be left for bonding pads to connect the machine to external memory or other wafer scale CHiP machines. But there is a more important and subtle reason for limiting the number of blocks on the wafer.

Defects, in general, are randomly distributed over the wafer surface. The yield model developed in Chapter 2 is based, in part, on this assumption. As a result, the analysis of fault tolerance, and subsequently, the choice of a 4 × 3 building block depends on random distribution of defects. This assumption applies quite accurately to the entire wafer except for its periphery [Stap73, Stap76, Laws66]. A band at the outer edge of the wafer exhibits a substantially higher density of defects [Gupt72]. This results from several processing effects:

[1] 5" wafers have been available for some time but are gaining acceptance slowly due to some incompatibilities with existing fabrication equipment.

a) crystal dislocations formed during crystal growth

b) nonuniform diffusion caused by temperature variations at the wafer periphery. This is particularly acute near the orientation flat that is in contact with the cooler diffusion boat.

c) beading of the photoresist near the edge

d) rounding of the wafer at the edge which causes pattern distortion.

The defect density measured inward from the edge decreases exponentially to a constant value for the central region of the wafer. The width of the region in which the density is significantly increased has been reported to be in the range 4-5 mm [Gupt72] although it can be expected to vary considerably from process to process.[1]

To accommodate these phenomena, building blocks are placed in the central portion of the wafer and bonding pads are located on the periphery (Figure 5.1.1). Pads are simply areas used as targets for soldering wires onto the silicon. Their functionality is unaffected by the presence of defects in the silicon. On the other hand, processing elements and switches are in general rendered dysfunctional by defects. Therefore they are located in the large central portion of the wafer where defects are fewer.

This results in efficient utilization of the wafer area. Instead of uniformly distributing processors and bonding pads over the wafer (as in a conventional layout), they are separated and placed in the most appropriate portion of the wafer. Although processing exhibits a great deal of variability,

[1] Industrial sources are very reluctant to reveal any exact figures regarding yield results. One source [Stap76] defines a two area model with "inner" and "outer" rings of the wafer exhibiting different defect densities, but the dimensions of the regions are not specified.

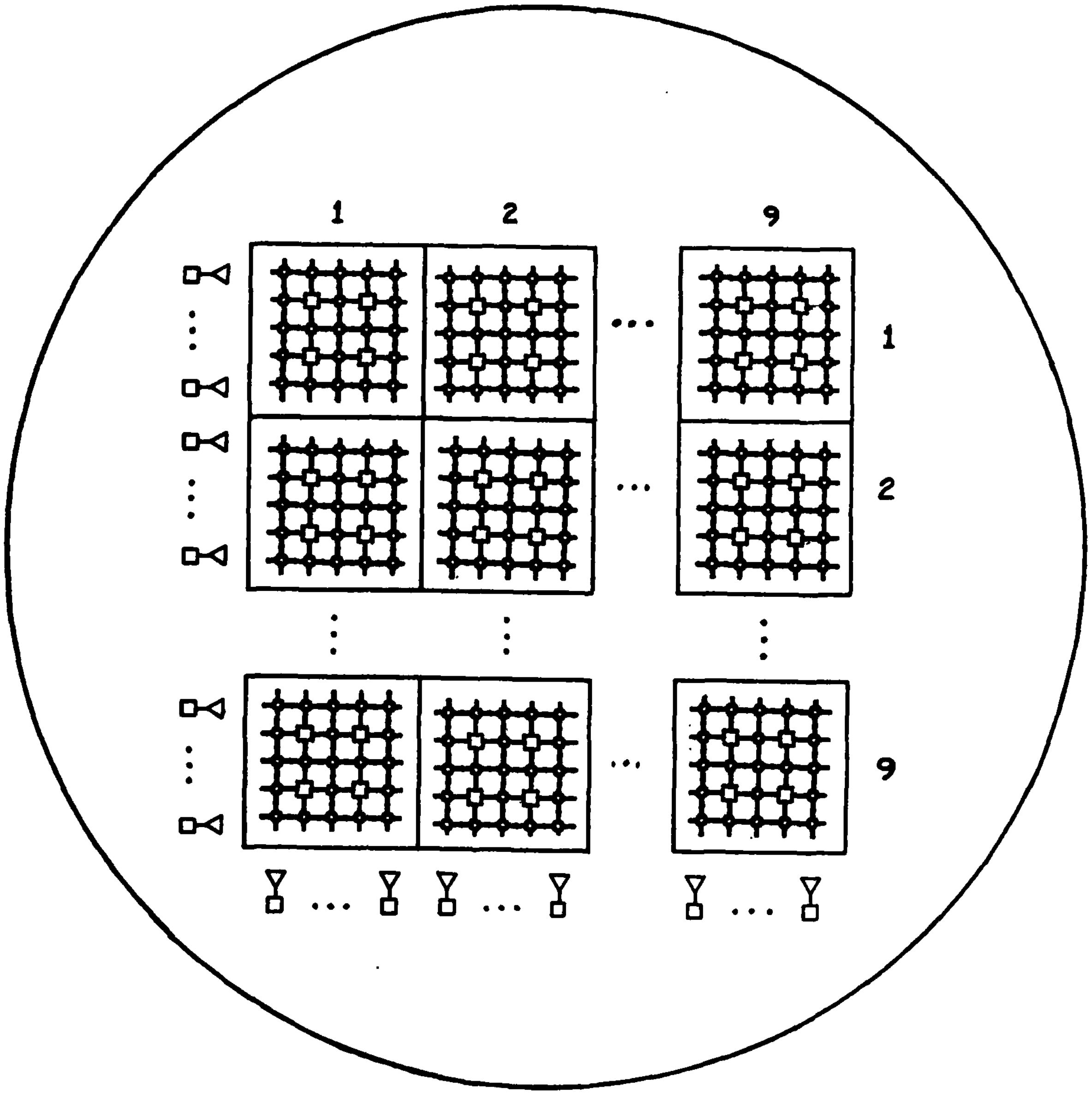


Figure 5.1.1 - Layout of a Wafer Scale CHiP Processor

some researchers report virtually no functional chips in the outermost 3 mm. The wafer scale machine effectively utilizes some of this area. In sum, defect insensitive components are placed where defects are most frequent, and defect sensitive circuitry is located where there are fewer defects.

2. Lattice Dimensions

The layout of the wafer scale CHiP processor is shown (in somewhat schematic form) in Figure 5.1.1. In the center of the wafer is an 8 × 8 grid of building blocks. From the results of the previous section, each of the blocks has a 99% probability of containing a fully functional 2 × 2 mesh. When a wafer contains a block that does not have a 2 × 2 mesh, the entire column containing the faulty block is discarded. The column exclusion strategy described in Chapter 1 is used to eliminate the occasional defective block.

On the average, how many usable blocks will a wafer yield? Since the defects are randomly distributed, the chances of the individual blocks being functional are independent events. Because the events are either "success" (*i.e.* functional) or "failure" (*i.e.* faulty), the probability distribution of good and bad blocks is a binomial random variable. Let Pr(F = i) represent the probability that exactly i blocks are faulty.

$$\Pr(F = i) = \binom{64}{i} (1 - p)^i \, p^{(64 - i)}$$

where p = 0.99 is the probability that a block is functional, a successful event. The probability of occurrence of a given lattice size is derived as follows:

Derivation of Table 5.2.1

The probability of having a completely functional 8 × 8 grid is simply the probability that there are no defective blocks, $(0.99)^{64} = 0.526$.

In general each defective block eliminates an entire column of blocks. But to accurately compute the probability of occurrence of a given lattice size, we must account for defective blocks falling in the same column of the grid. In this case, only a single column is lost despite the occurrence of multiple defects.

The probability of exactly one excluded column (giving a 16 by 14 lattice since each column is two PEs wide) is:

$$\Pr(F = 1) + \Pr(F = 2)\Pr(\text{2 bad blocks in same row or col}) =$$
$$0.340 + (0.108)\frac{14}{63} = 0.364$$

The first defective block can occur anywhere in the grid. There are seven blocks in the same row as the first defective block and seven in the same column. So 14 of the remaining 63 blocks can be faulty but still leave just one row (or column) excluded. The chances of 3 or more bad blocks occurring and all falling in the same column are negligible.

The probability of exactly two excluded columns (yielding a 14 × 14 lattice) is similarly derived:

$$\Pr(F = 2)\Pr(\text{2 bad blocks fall in different cols}) +$$

$$\Pr(F = 3)\Pr(\text{3 blocks occupy exactly 2 cols}) =$$

$$(0.108)\ \frac{49}{63} + (0.0226)\left[\frac{16}{80} + \frac{64}{80}\,\frac{50}{79}\right] = 0.0914.$$

Table 5.2.1 shows the different possible grid sizes resulting from an 8 × 8 grid on a wafer and their probabilities of occurrence. About 53% of the time all blocks are usable, and the wafer holds a CHiP processor of size 16 PEs by 16 PEs. 36% of the wafers contain exactly one excluded column. With each block being 2 PEs wide, a 16 × 14 PE processor is recovered from the wafer. Only 1.9% of the wafers will yield a CHiP machine of size smaller than 14 × 14. The expected number of usable PEs per wafer is 237. This represents a truly large-scale parallel processor on a single wafer, and this is achievable with current technology. With future scaling of device dimensions, even more processors per wafer will be possible. Thus, these results indicate that the processing element portion of a parallel processing system can indeed be constructed from a small number (perhaps one) of wafer scale components.

The choice of an 8 × 8 grid is quite conservative. It results in substantial wafer are being left for bonding pads and drivers or to be unused due to high defect density. In fact, an 8 × 8 grid occupies area

$$64 \times (8.23)^2 = 4335 \text{ mm}^2$$

(recall that each building block has an edge length of 8.23 mm, see section 1). But a 4" wafer has area 8107 mm^2 so only 53% of the wafer is occupied by the CHiP lattice. Why was the 8 × 8 grid proposed? For the simple reason that it is a safe choice. It is the largest square lattice that fits onto a 4"

Table 5.2.1 - Size of Wafer Scale Processor for 8 × 8 Grid

Lattice Size from an 8 × 8 Grid			
probability	cumulative probability	grid size	size of CHiP processor (PEs)
.526	.526	8 × 8	16 × 16 = 256
.364	.890	8 × 7	16 × 14 = 224
.0914	.981	7 × 7	14 × 14 = 196
.0186	1.000	< 7 × 7	

Expected Number of Good PEs = 237

Table 5.2.2 - Wafer Area Occupied by a Grid of Building Blocks

grid size	area (sq mm)	% of 4" wafer area
8 × 8	4335	53.5
9 × 8	4877	60.2
9 × 9	5486	67.7
10 × 9	6096	75.2
10 × 10	6773	83.5

Table 5.2.3 - Size of Wafer Scale Processor for 9 × 8 Grid

Lattice Size from an 9 × 8 Grid			
probability	cumulative probability	grid size	size of CHiP processor (PEs)
.485	.485	9 × 8	18 × 16 = 288
.380	.865	8 × 8	16 × 16 = 256
.109	.974	8 × 7	16 × 14 = 224
.0199	.994	7 × 7	14 × 14 = 196
.0060	1.000	< 7 × 7	

Expected Number of Good PEs = 266

Table 5.2.4 - Size of Wafer Scale Processor for 9 × 9 Grid

Lattice Size from a 9 × 9 Grid			
probability	cumulative probability	grid size	size of CHiP processor (PEs)
.443	.443	9 × 9	18 × 18 = 324
.394	.837	9 × 8	18 × 16 = 288
.129	.966	8 × 8	16 × 16 = 256
.0271	.993	8 × 7	16 × 14 = 224
.0069	1.000	< 8 × 7	

Expected Number of Good PEs = 297

Table 5.2.5 - Size of Wafer Scale Processor for 10 × 9 Grid

Lattice Size from a 10 × 9 Grid			
probability	cumulative probability	grid size	size of CHiP processor (PEs)
.405	.405	10 × 9	20 × 18 = 360
.400	.805	9 × 9	18 × 18 = 324
.140	.945	9 × 8	18 × 16 = 288
.0414	.986	8 × 8	16 × 16 = 256
.0139	1.000	< 8 × 8	

Expected Number of Good PEs = 329

wafer with a ***substantial*** safety margin of area. This area is required for bonding pads, drivers, regions to be unused due to high defect density, area loss due to the packing of rectangular blocks, the wafer's orientation flat, variations from fabrication process to fabrication process in the size of a unit area, etc. In accordance with Slotnik's Law, in this section the machine architecture proposed incorporates as few new features, in addition to wafer scale integration, as possible. Highly conservative choices are made for virtually all design decisions. Additionally, variances from the conservative choices are noted and their effects are analyzed.

The 47% unused area in the 8×8 grid is a very large safety margin. It is quite likely that larger grids can be accommodated on a 4" wafer. (Or alternatively, one could fabricate an 8×8 grid with larger PEs that are more complex and faster. This option is more complex to analyze since changing the PE area necessitates a reexamination of the degree of redundancy required within a block. A 4×3 block may not be appropriate for substantially larger or smaller PEs.) The maximum size grid that can be patterned on a wafer depends on the details of the fabrication process, layout details of the processing elements and switches, and wafer characteristics. This must be determined experimentally for a particular combination of PE design and process technology. We will be content to propose a conservative approach and note the extensions that may be possible.

Consider the range of possible grid sizes. First, what is the upper bound on the wafer area that can be occupied by a grid? Once again, this is strongly dependent on the particular technology, but we make some rough estimates. Assume the outermost 5 mm of the wafer is unusable due to high defect density. The ring of pads and drivers is approximately 0.2 mm wide. To make a conservative estimate of the effective area, assume that the bonding pads are placed within the 5mm outer ring. This will define a lower bound on the effective wafer diameter. Thus the effective wafer diameter is reduced from 4" (101.6 mm) to 91.2 mm. The area of this central portion of the wafer is 6532 mm^2 or 80.6% of the total wafer area. Table 5.2.2 shows the area occupied by grids of different dimensions. A 10 × 9 grid is the maximum allowed by the above bound.

Consider a possible alternative to the 8 × 8 grid. A 9 × 9 grid leaves 32.3% of the wafer area unused. This constitutes a fairly large safety margin. It is still well below the 80% bound on usable wafer area derived above. Thus a 9 × 9 grid is a reasonable choice for a 4" wafer although it pushes the limits of technology more than the conservative 8 × 8 grid.

With the 9 × 9 grid, 44% of the wafers will have no excluded columns and will realize a 18 PE by 18 PE processor (Table 5.2.4). This is a truly large parallel machine. It represents a 25% increase over the 8 × 8 grid. Another 39% of all wafers will have exactly one defective block and will implement an 18 × 16 processor array. This is still 12.5% larger than the maximum size machine achievable with the 8 × 8 grid. In total, 96% of the wafers will host a CHiP processor at least as large as 16 PEs by 16PEs. The expected number of good PEs per wafer is 297. This is 27% more than the 8 × 8 grid. In summary, a substantially larger CHiP lattice is obtained with a 9 × 9 grid as compared to an 8 × 8 grid.

What effect does the use of a larger grid have on the size of the CHiP machine? Tables 5.2.3 - 5.2.5 show the lattice sizes obtainable with grids larger than 8 × 8. The expected number of good blocks per wafer increases in direct proportion to the grid size. As the grid size increases, the probability of a fully functional grid decreases from ~50% to ~40%. With more building blocks, there is an increased chance that one block will be faulty. With technological improvements, the size of PE's and switches will continue to decrease thus making even larger grids possible. The increased possibility of a faulty block may ultimately put a limit on the maximum grid dimensions.

3. Column Exclusion

When a column (or row) contains a faulty block and is excluded, the adjacent columns must be connected together. The switches and datapaths in the unused or faulty blocks are used to make the connection. Thus the "wire around" requirement for blocks becomes a "wire through" capability via the CHiP switch lattice (Figure 4.1.1). The double corridor width switch lattice of the building block provides twice as much wiring bandwidth through the lattice as is necessary. This redundant wiring capability can be used to circumvent faulty switches. As shown in the previous chapter, blocks provide wire through capability with very high reliability.

However, each switch introduces additional signal delay since a signal must pass through a pair of transfer gates in each switch. To traverse an unused column, typically ten to fourteen extra switches are introduced into the path. In addition to switching delays, this requires that periodically in the path the signal must be boosted by a super buffer to prevent catastrophic signal degradation. But buffers introduce additional delays. In short, column exclusion incurs a performance penalty.

The amount of signal delay incurred depends on the impedance of the individual switches and the number of switches separating PEs. The design of low impedance switches is an important practical problem in the implementation of the CHiP family of machines. A combination of circuit design and fabrication technology techniques such as the use of depletion mode transfer gates with high channel doping levels reduces impedance. These techniques substantially reduce switch delays. However, the delay through even a fast switch is more than the delay incurred by directly wiring together processors. The gain in flexibility due to the switch lattice is bought at a loss in performance. This problem is common to all machines in the CHiP family.

The number of switches between two PEs depends on two interrelated factors: the specific PE configuration and the corridor width of the switch lattice. The processor configuration is under the control of the programmer. Some topologies can be mapped onto a lattice efficiently with only short distances separating the PEs (for example, the mesh). Other more complex arrangements require longer paths. A wider corridor width provides additional wiring bandwidth and will in some instances allow more compact layouts. But in any event, the corridor width of the switch lattice is the minimum separation for any configuration. Since wafer scale systems must be robust to switch failures in addition to processor faults, they must have extra switching corridors used exclusively for fault tolerant reconfiguration. Thus, wafer scale systems, with their redundant switches, increase the number of switches that inter-PE signals must traverse. Wafer scale systems pay for their low cost in the currency of performance.

4. External Connections

Consider the requirements of connecting the wafer scale machine to external devices - either memory or other CHiP machines or both. At the very maximum, every switch on the lattice edge has an external connection.[1] With a data transfer width of one byte and two control lines per datapath, each switch requires ten bonding pads. In a 16 PE by 16 PE lattice there are 32 switches on a lattice edge or 320 bonding pads per edge. (No external connections need be provided for the redundant switching corridors since they are used exclusively for fault tolerant reconfiguration.) Each bonding pad is a square with edge approximately 0.1 mm on a side and 0.075 mm spacing between pads [Mead80]. So at a total width requirement of 0.175 mm per pad, a line of 320 pads extends 56 mm. This is just slightly more than the radius of the wafer. Counting all lattice edges, 4 radii of pads are required. The circumference of the wafer is 2π radii long. So the pads can be arranged around the perimeter of the wafer in a single circular pattern. Note that additional external connections can be implemented with multiple concentric circles of pads.

The off-chip drivers are located between the circle of bonding pads and the CHiP lattice. They connect a subset of the switches on the lattice edge to bonding pads and provide the required signal amplification to reliably and

[1] In practice, providing connections just for the switches directly connected to PEs should be sufficient to meet the I/O requirements. This cuts the number of external connections at least in half which may be more in line with the limitations of packaging technology. The above represents a worst case analysis.

quickly transmit signals to an off-chip source.

A CHiP machine can not afford to have a switch on the lattice edge with a missing external connection. The interface of the switch lattice to its external connections must be complete and symmetric. Therefore the integrity of the driver circuitry and the connections to the bonding pads and switch lattice must be very high. There is the potential for the loss of an entire column of blocks should a driver fail.

A number of steps can be taken to insure reliability. First, the drivers are placed inside the band of high defect density near the wafer edge. This removes them from the wafer area most prone to circuit faults. The exact location depends on the wafer characteristics and the sensitivity of driver circuitry to defects.

Second, drivers can be designed to be highly reliable. Pad drivers are relatively simple which reduces their chance of failure. Also, much of the circuitry is composed of large transistors - many times the size of a minimum geometry transistor [Hon80]. This is necessary due to the large power and current requirements of off-chip signals. Large size decreases the sensitivity to defects and increases yield. Additionally, the entire pad driver, especially the smaller geometry circuitry on the switch lattice side, can be designed with relaxed design rules. This once again can substantially increase yield. Wider wires with larger spacings are less likely to fail. This slightly increases the pad area and the signal transmission time but is a small price to pay for increased reliability.

Third, provide redundant drivers. In addition to making drivers reliable, add 100% driver redundancy at each pad. In the rare case that a driver is faulty, its redundant counterpart functions in its place. Both drivers are connected to the pad (and switch) via a common bus (see Figure 5.4.1). In case of an active fault (*e.g.* a short of the bus connection to Vdd or Gnd), the driver can be physically disconnected from the bus by laser trimming or fuse blowing. The bus wire can be made wide enough and with sufficient spacing from neighboring circuitry to insure bus integrity. Lastly, redundancy in the form of complete pad / driver combinations can be added. This guards against the occurrence of non-random defects at the wafer edge.

In summary, the problem of providing extremely reliable off-chip drivers can be solved by technological means. There are no fundamental difficulties. A combination of driver reliability achieved through relaxed design rules and redundancy achieves the required reliability. The exact combination of these techniques required to produce the desired reliability is technology specific.

5. Efficiency

In each block only four of the twelve processors are used regardless of how many more are actually functional. Furthermore, every time there is one bad block, an entire column of eight blocks is discarded. It appears that the two level hierarchy approach to implementing wafer scale integration makes very inefficient use of the wafer surface. Surprisingly enough, this is not the case.

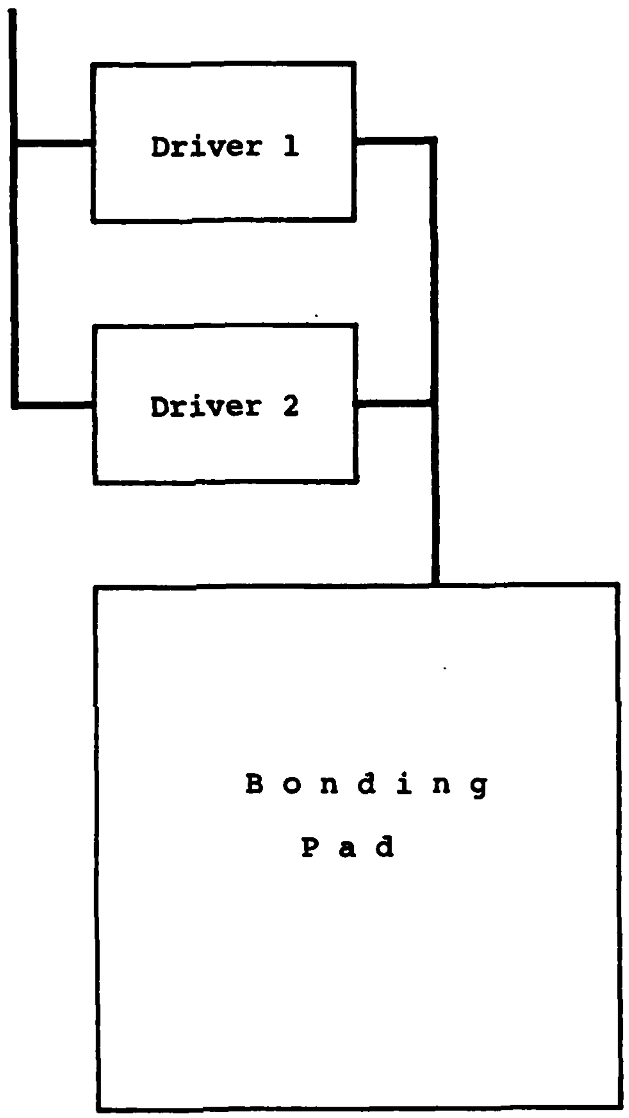


Figure 5.4.1 - Redundant Pad Drivers for High Reliability

Consider the alternative to implementing a 2×2 lattice with fault tolerant building blocks. The fault tolerant approach will be compared to conventional manufacturing of integrated circuits without redundant components. Let us simply pattern as many 2×2 lattices on the wafer as possible, scribe the wafer into the individual 2×2 lattices and package them. Since the 2×2 lattices are considerably smaller than the fault tolerant building blocks, a 4" wafer can hold 321 of them. At 20% yield (our reference point since one normalized unit area holds a 2×2 lattice and has by definition 20% yield), there are $321 \times 0.20 = 64$ good lattices per wafer. In the wafer scale machine (with the conservative choice of an 8×8 grid), the expected number of PEs is 237 occurring in 59 2×2 lattices. This is 9% fewer than with conventional processing.

Is this a victory for the conventional approach? Not quite. First, the number of 2×2 lattices actually patterned on the wafer will be lower than 321. The bonding pads required at each lattice have not been accounted for. As a result, the area of each lattice must be slightly larger.[1] Also there must be scribe lines between lattices. This consumes a little more area leaving less for the lattices. Secondly, the increased defect density along the edge of the wafer greatly reduces the chip yield there. 20% yield is achieved only in the central portion of the wafer. Averaged over the entire wafer, somewhat less than 20% yield will actually be realized. As a consequence, there will be fewer than 64 good lattices per wafer with

[1] One advantage of the wafer scale approach is that there are fewer total number of bonding pads. The internal lattice connections are made not by large (and slow) pads and off-chip drivers, but by direct wiring in silicon from PE to PE.

conventional processing. The exact number depends on processing details.

Just as we were liberal with the estimates in the conventional approach, we have been conservative in the estimations for the wafer scale case. Remember that the 8 × 8 grid of building blocks is a very conservative choice that occupies only 53% of the wafer area. In practice, a larger grid could be used. The exact dimensions of the largest lattice that can be patterned on a wafer is dependent on the particular fabrication process and the characteristics of the wafers. This must be determined experimentally, but in any case, there would probably be more than 59 lattices per wafer.

In short, the initial estimates overstated the number of good lattices obtained through conventional technology and understated them in the wafer scale case. In practice, the number of good lattices per wafer is comparable in both approaches, but the exact numbers of good lattices is dependent on processing technology. As a result, we can conclude that the

- use of fault tolerant building blocks to implement wafer scale integration makes efficient use of silicon area.

The reason behind this is that the area lost to redundant PEs is more than made up for by the increased yield provided by the redundancy. Examining the curve of building block recovery vs. the number of PEs (for the recovery of a fixed size 2 × 2 lattice, Figure 2.5.2), we find the curve rises quite quickly This means a small amount of fault tolerance has a big payoff in terms of yield; a modest amount of fault tolerance has high leverage.

On the other hand, the area increase due to redundancy is linear. The area of a lattice increases in direct proportion to the number of processors. This follows for two reasons. First, the mesh connected structure of the lattice requires that each component be connected to only a fixed number of other components. The number of connections does not increase with the size of the lattice. (This property is not enjoyed by many of the other interconnection networks. For example, the binary cube requires that each processor in an N node machine be connected to $\lceil \log N \rceil$ other nodes. Thus the number of wires per processor can be very large for large-scale binary cubes.) Secondly, the local connection structure of the mesh requires that each node be connected only to its physically adjacent neighbors. Each of the wires connecting PEs has constant length independent of the lattice dimensions. The distance of a PE from its neighbors to which it is connected is independent of the size of the mesh. (Once again few other interconnection strategies preserve locality. A perfect shuffle connection network has a constant number of connections per processor regardless of the network size. But each node must be connected to a node in a fixed *relative* position in the shuffle. For example, node 1 is connected to node N/2. So, as N increases, the length of each connection (on the average) increases. As a result, a perfect shuffle of N PEs requires $O(N^2 / \log^2 N)$ area [Klei81].) With the number of wires and their length both constant, the area occupied by wires increases in proportion to the number of processors in the mesh. Since PE area is also independent of lattice dimensions, the lattice area grows linearly with the number of PEs.

As was shown in Chapter 2, redundancy can provide large increases in the recovery probability. This means that modest amounts of redundancy increase the efficiency of use of the wafer area. The area taken up by redundant PEs is more than made up for by the increased recovery. In Chapter 2 it was seen that modest amounts of redundancy (*e.g.* ~50%) lead to optimum use of the wafer area. The need for very high block yield (as required by the column exclusion strategy) necessitates that building blocks have much higher redundancy than for optimal area utilization. However, the PE utilization does not fall below the PE utilization without redundancy. Utilization for conventional, non-redundant chips and building blocks are similar.

6. Effect of Technological Advances

The wafer scale CHiP machine described above can be fabricated with current (1982) technology. Four inch wafers are the industry standard and have been commonplace for several years. The complexity of the processing elements is less than that of a simple microprocessor, and switches are considerably simpler. The design of the individual components is straight forward in comparison to the current generation of advanced microprocessors. Simple PEs 1.75 mm on a side can be produced with gate lengths and wire widths attainable by current state of the art semiconductor manufacturing processes. In summary, the wafer scale processor does not depend upon unconventional or experimental technology.

However, semiconductor fabrication technology is not static. Transistors will continue to shrink in size. Defect densities will continue to be reduced. Wafers will become purer and larger in diameter. In short, more circuitry will continue to be packed into a smaller area with decreased power consumption and increased circuit speed. The pace of these advances has been slowing in recent years due to increasingly difficult technological problems, physical limitations and mounting capital costs of the increasingly sophisticated fabrication equipment. Although the pace of advancement is slowing, the trend is inexorable [Noyc77].

What will be the effect of technological advances on wafer scale machines? Larger wafers will allow the fabrication of CHiP lattices of larger dimension which are composed of more powerful processing elements. Also, the scaling down of device sizes has positive impact on virtually all circuit parameters. Processing elements will become smaller, more reliable and less power hungry. This will lead to larger lattices on the wafer, less redundancy required within each building block and reduced switching overhead. Although the direction of these trends is clear, this section *quantitatively* analyzes the effect of technology improvements on wafer scale CHiP processors.

In previous sections, the estimates of PE size and number of PEs per wafer are based on a conservative assessment of current technology. We have assumed 4" wafers and transistors with 2 μm channel lengths. Both of these are typical of state of the art fabrication processes currently (1982) in volume production. This represents the baseline case against which technological advances will be compared. For the purposes of comparison, we will project a short term and a long term technological advancement. Some major facets of the design of wafer scale CHiP processors will be reconsidered in these new contexts and compared to the baseline case.

a) Wafer Size

Wafer diameter has steadily increased over the years. In the early 1960s, wafers 1.5" in diameter were common. Today, 4" wafers have be commonplace for some years. They are the standard of the industry. Additionally, 5" wafers are available. Due to some incompatibilities with existing fabrication equipment, their use has not become widespread but their acceptance is growing. A fivefold increase in wafer area over the span of two decades is a snails pace compared to the pace advances in device scaling. Consequently, as the representative of long term future technology, a modest increase in wafer diameter to 7" is selected. This represents a doubling of the area of the current state of the art 5" wafer.

In the following discussion, the characteristics of wafer scale machines fabricated on 5" and 7" wafers will be compared to 4" wafer. The 4" wafer represents the baseline case for well established current technology. 5" wafers are at the cutting edge of the current state of the art, and 7" wafers represent the possibilities of long term future technology.

The characteristics of wafers with diameter of 4", 5" and 7" are shown in Table 5.6.1. The 7" wafer has over three time the area of the 4" wafer. However, recall that not all the wafer area can be occupied by building blocks. Assume that the outer 5mm of a wafer can not be occupied by building blocks due to high defect density. (In practice, some of this area will be occupied by bonding pads and their drivers.) We will estimate the number of building blocks that can be fabricated on a wafer. Define the effective diameter of the wafer as the wafer diameter minus 10mm. It delimits a lower bound on the *effective wafer area*. This is the area that can potentially be occupied by processing elements and switches.

Table 5.6.1 - Effect of Wafer Diameter on the Wafer Scale CHiP Processor

	Wafer Diameter 4"	5"	7"
total wafer area (sq mm)	8107	12,668	24,829
effective wafer area (sq mm)	6590	10,751	22,114
maximum number of blocks in effective area (lower bound)	77.6	133.6	290.4
maximum grid size (blocks)	9 × 9	11 × 11	17 × 17

The increases in effective area between the 4" wafer and the larger ones are even more pronounced than the increase in the total area. Removing a fixed size outer band eliminates proportionately more area from small wafers than from large ones. The effective area of a 5" wafer is 63% larger than the 4" wafer, and the 7" wafer has 3.4 time the effective area of the 4" wafer. There is room for substantially more building blocks on the larger wafer.

The maximum number of square building blocks with edge length e that can be packed onto a circular wafer of diameter D is given by formula 1.1

$$\frac{\pi D^2}{4e^2} - 1.77\,\frac{D}{e}$$

Using the above equation, Table 5.6.1 shows that the maximum number of blocks per wafer increases by 72% for the 5" and 274% for the 7" wafer. As expected, much larger CHiP processors can potentially be fabricated on the larger wafers. The maximum number of blocks increases more quickly than the effective area. Note that the effective area increases are only 63% and 236% for 5" and 7" wafers respectively. The reason for this is that larger wafers are less effected by edge curvature. With an arc of larger radius, the relatively small building blocks can be placed around the the wafer edge with less waste of area. Additionally, with larger wafers, a larger fraction of the wafer area falls in the center and is unaffected by edge curvature. In particular, from the second term of equation 1.1 we see that a 4" wafer has a 20% reduction in the number of blocks due to edge curvature. 5" and 7" wafers lose only 16% and 11% of their blocks respectively. In summary, building blocks can be packed more efficiently into larger wafers than smaller diameter wafers. This results in more efficient use of the wafer area for larger diameter wafers. The increase in the size of a wafer scale CHiP processor that can be patterned on a larger wafer is greater than simply the increase in wafer area. A 7" wafer can hold 3.7 (= 290.4 / 77.6) as many standard PEs as a 4" wafer whereas the ratio of total wafer area is only 3:1.

In terms of maximum square grid size, a 4" wafer can hold a 9×9 grid. An 11×11 fits onto a 5" wafer, and a 7" wafer can hold a huge 17×17 grid of building blocks. This represents a 34 PE by 34 PE CHiP lattice - a truly large-scale parallel machine. Even the use of a 5" wafer (which is well within the scope of current technology) allows the fabrication of a CHiP lattice with 50% ($\approx 11^2 / 9^2 - 1$) more PEs than the 4" wafer. In summary, even a modest increase in wafer diameter substantially increases the maximum size of a wafer scale CHiP processor through both an increase in wafer area and more efficient utilization of that area.

b) Device Scaling

As advances in semiconductor manufacturing technology continue, the size of devices continues to be reduced. Wires become narrower and transistors smaller. Although the rate of progress is slowing, further advances can be expected. What will be the effect on wafer scale machines? This section examines some of the consequences of smaller processing elements and switches on the design of wafer scale CHiP processors.

In the previous sections, the area estimates for PEs and switches were based on Mead and Conway design rules under the assumption that $\lambda = 1$ μm. This corresponds to a transistor channel length of 2 μm and is conservatively representative of current technology. Intel's HMOSII process achieves 2 μm channel length and has been in volume production for several years. HMOSII is a mature technology and its successor will soon be introduced. As a result, we select as a representative of near term technology a doubling of the device density. This corresponds to shrinking the lateral dimensions of devices to 70% of their current dimensions - a channel length of approximately 1.4 μm. As for the long term advances in device scaling, the DOD has launched a concentrated effort to achieve 1 μm technology which would quadruple the device density. It appears that this goal is achievable through the extension of current optical lithography techniques, and it is a feasible goal for the late 1980s. Consequently, a channel length of 1 μm is selected as the representative of long term technology advances.

With smaller PEs, their yield increases. When the PE area is shrunk in half, the yield (computed by the yield model, equation 2.2) for a 2 × 2 CHiP lattice doubles (Table 5.6.1). Higher PE yield reduces the amount of redundancy that is required to achieve 99% block yield. Consider the reduction in device area by a factor of two. Examining Figure 2.5.4 shows that four functional standard PEs (with their area scaled by a factor of 0.5) can be found in a group of 9 PEs with 99% probability. Thus the dimensions of a building block with a 99% recovery rate can be reduced from 4 × 3 to 3 × 3. Only five redundant PEs per block are required instead of eight. Redundancy is decreased by one third with no decrease in the recovery probability of the block. Not only is the area of a single PE cut in half, but the number of PEs in a block is reduced. This results in a double area savings - smaller PEs and fewer of them.

There are two main consequences of the reduction in block dimensions due to smaller PE size:

1) More efficient use is made of the wafer area. Wafer scale integration implemented via column exclusion imposes overhead in the form of the redundancy required to achieve high block yield. The redundant PEs are not an integral part of the CHiP lattice; only a fixed number of PEs are recovered from each block. But still they occupy area that could be used by the lattice. Smaller PEs have higher yield and require less redundancy for the same block recovery rate. This frees wafer area for additional blocks.

2) Smaller building blocks have shorter paths between PEs. Recall that the maximum path length between two PEs is determined by the building block dimensions (Chapter 4). Reducing the block size to 3 × 3 results in fewer switches between PEs and decreased signal transmission time. Device scaling leads to not only more efficient use of the wafer area, but decreased switching overhead. Performance is correspondingly enhanced.

How much can the block area be reduced by the use of the smaller PEs and switches? In the baseline case, each 4 × 3 building block occupies a 67.7 mm^2 region of silicon (Table 5.6.2). By scaling down this value, we estimate that the area occupied by each 3 × 3 building block to be approximately

$$\frac{67.7}{2}\,\frac{9}{12} = 25.4 \ (mm^2)$$

Table 5.6.2 - Effect of Device Scaling on the Wafer Scale CHiP Processor

	Relative Area		
	1.0	0.50	0.25
channel length (μm)	2.00	1.41	1.00
PE area (M μm **2)	12.3	6.13	3.06
yield of a 2 × 2 lattice	0.200	0.412	0.627
PEs / block for 99% Recovery	12	9	7
building block area (sq mm)	67.7	25.4	9.88
block edge length (mm)	8.23	5.04	3.14
grid size per 4" wafer	9 × 9	14 × 14	23 × 23

The area of each PE and switch is cut in half, and the number of PEs is reduced from twelve to nine. Assuming a square block, the block edge length is $\sqrt{25.4} = 5.04$ (mm).

How many of these smaller building blocks can be placed on a wafer? As shown previously, a 9 × 9 grid of blocks with edge length 8.23 mm can be fabricated on a single 4" wafer. In the scaled down technology, the shorter block edge length means that a grid of roughly

$$\frac{8.23}{5.04} 9 = 14.7$$

blocks per side can be fabricated on a 4" wafer. Rounding this down, the wafer can hold a 14 × 14 grid of building blocks. Since the same 2 × 2 virtual lattice is mapped into each of the building blocks, a 28 PE × 28 PE lattice will fit on a single wafer. The number of PEs increases by a factor of 2.4 (= 28^2 / 18^2). So cutting PE area in half more than doubles the number of PEs per wafer due to the increased efficiency in the use of the wafer area. There is an additionally 20% (= 0.4 / 2.0) increase attributable to increased efficiency. Note that the increase due to efficiency is not equal to the reduction in the number of PEs per block, 25% = (12 - 9) / 12. The increase is lower due to the restriction that the wafer contains a square grid of building blocks. If $(14.7)^2$ blocks could be put on the wafer, a full 25% gain due to efficiency would be realized.

Now consider the quadrupling of the device density. The yield of a 2 × 2 lattice of standard PEs more than triples to 62.7%. Once again the yield increase reduces the amount of redundancy required. Only three extra PEs are required to give a 99% recovery rate of four PEs (Figure 2.5.5). Redundancy is reduced from eight extra PEs in the baseline case to only three PEs. The building block area is correspondingly reduced to

$$\frac{67.7}{4}\frac{7}{12} = 9.87 \text{ (mm}^2\text{)}$$

with the block edge length of $\sqrt{9.87}$ = 3.14mm. This results in a grid of no more than

$$\frac{8.23}{3.14}9 = 23.6$$

blocks per side (Table 5.6.1). This is an increase of 653% over the baseline case. Of this, 400% is directly attributable to smaller PEs, and the remainder to the reduction in the number of PEs per block.

7. Practical Implementation Considerations

The previous chapters have covered the general principles of the implementation of wafer scale integration: two level hierarchy, column exclusion and fault tolerant building blocks. Structuring, the major hurdle in the implementation of wafer scale systems, is achieved through a combination of these design principles. In addition, a number of lower level implementation issues have also been discussed: wafer layout, switch lattice structure, external connections, etc. Despite the (apparent) success of this approach, a host of engineering problems must all be solved before the wafer scale CHiP machine can make the transition from paper to silicon. The problems of heat dissipation, clock skew, routing of power and ground wires, etc. must be addressed before a wafer scale machine can be constructed. A number of these practical implementation considerations are discussed in this section.

a) Power Consumption

Electrical signals are changed by the storing and discharging of electrical energy. This requires the application of power which is transformed into heat. To maintain a continuously operating device at an acceptable temperature, this heat must be transferred from the device to the surrounding environment. As more and more devices are packed into smaller and smaller volumes, there is a greater concentration of heat in a smaller volume with less surface area available to conduct away the heat. Power dissipation becomes increasingly difficult. The problem of power dissipation is a difficult one for high density LSI chips.

This problem is particularly acute for wafer scale systems. A wafer scale system has on the order of 100 times as many components as a complex LSI chip. This very large number of circuits is packed into a single package. A single wafer scale system may replace an entire printed circuit board resulting in a large increase in the density of gates per cubic centimeter.

To address this problem, we will first estimate how much power can be dissipated by a wafer. This will in turn dictate the power consumption requirements of the individual switches and processing elements. Finally, the design of the switches and PEs to meet these power requirements will be considered.

Since we have not proposed one specific design and layout of PEs and switches, the power consumption figures derived below will necessarily be rough estimates. Exact figures can be obtained only for a specific processor. We will attempt to show that the class of wafer scale CHiP processor discussed in this work can with proper design meet reasonable power consumption restrictions.

A single chip can dissipate 1W with only common and inexpensive packaging technology. Up to 5W per chip can be dissipated through the use of exotic and expensive packaging techniques such as direct water cooling, heat sinks and cooling towers. A wafer has approximately 200 times the surface area of a single chip; there is a much larger surface area over which to perform the heat exchange with the surrounding environment. With similar packaging technology, the larger wafer scale system should be capable of dissipating substantially more power than a single chip.

With forced air cooling a printed circuit board can dissipate up to approximately 0.5W per in^2 [Stee81]. With the surface area of a 4" wafer being 12.6 in^2, this indicates a limit of approximately 6W per wafer. Since the 0.5W / in^2 figure was for printed circuit boards consisting of a number of separate packages, the application of this estimate to a single package wafer scale systems may not be entirely accurate. Consequently, 6W will be regarded as an upper bound. In accordance with our conservative design philosophy, in the following considerations we will attempt to not exceed 50% of this bound. 3W per wafer will be the target for power consumption.

CMOS technology is the natural choice for reducing power consumption [Yu81]. The speed-power product for CMOS gates is substantially lower than for any other technology. CMOS circuitry typically runs at a small fraction of the power consumption of an identical circuit implemented in nMOS technology.

An additional advantage of CMOS technology is that the static power consumption of gates is virtually zero. CMOS gates consume power only when they are changing state. A static gate draws only the current necessary for the gate leakage current - on the order of a few nanoamps. On the other hand, with nMOS circuitry, all gates that are "on" continuously draw an appreciable amount of power.

This is especially advantageous for CHiP processors since they have a large number of static components. The switch lattice structure remains fixed for relatively long periods of time. The switch settings remain unchanged except when the lattice is being reconfigured into a new interconnection pattern. With CMOS implementation, the switches will draw essentially no power except during a reconfiguration. Since there are a very large number of switches on a wafer (~ 20,000), this results in a large power savings.

Furthermore, with CMOS technology the power consumption is directly proportional to the clock rate. The faster the gates change state, the more power that is consumed. This allows the system architect to fine tune the power consumption by varying the clock rate. System speed can be traded for power, if necessary.

As a result of the overwhelming advantage with regard to power consumption and the competitive speed and density characteristics of state of the art CMOS processing technology, it is proposed that

- wafer scale systems be implemented in CMOS technology.

Use of CMOS technology solves the power consumption problem (as will be shown in this section), but it introduces another difficulty. The estimates of PE and switch size (Chapter 4) were based on the implementation of the standard PE in nMOS technology. Implementing an identical design in another technology will not necessarily result in the layout occupying the same area. CMOS circuits typically require somewhat more area than their nMOS counterparts. As a result, a second pass through the design of building blocks (Chapter 4) should be made for the CMOS implementation of the standard PE.

However, state of the art CMOS processes require only marginally more area (*e.g.* ~ 10 - 15%) than the corresponding nMOS circuits and in some cases require slightly less area. Consequently, the CMOS area estimates depend on the particular design rules of the CMOS process and the details of the PE design, but in any case, the design of a building block should not vary drastically from that which was proposed in Chapter 4.

What power consumption requirements are imposed on the individual PEs and switches by the need to collectively dissipate a total of 3W? The answer to this question depends on the operation performed by the CHiP machine. CHiP processors operate in one of two modes:

a) Computational - the switch lattice is held in a fixed structure. The PEs compute and exchange data values.

b) Restructuring - during a restructuring phase, computation is generally not performed by the PEs, but rather the structure of the switch lattice is altered to provide a new interconnection topology. The individual switches each fetch a new current configuration setting from their local memory.

In a restructuring phase, how much power is consumed by the switches simultaneously accessing their local memories? To estimate this, we draw on power consumption figures for available memory chips. Recently announced 64K static RAMS implemented in CMOS technology have a power consumption of 10 μW in standby mode and 15 - 200 mW in active mode [Mina82, Koni82]. The local memory of a switch is in active mode when it is changing its current configuration setting. When not reading or writing, the memory is in standby mode. To estimate the power consumption of the switches, the above power consumption values will be scaled down in accordance with the size of the switch's local memory.

The PEs are quiescent during reconfiguration. The only power consumed by a PE is to maintain its local memory in standby mode. The maximum number of good PEs per wafer is 972 (= 81 × 12). With a 64 byte PE memory, the standby power consumption of the PEs does not exceed

$$972 \times \frac{64 \times 8}{65{,}536} \times 10\ (\mu W) \approx 75\ (\mu W)$$

The PEs consume a negligible amount of power during reconfiguration.

Now to estimate the power consumed by the switches during reconfiguration, note that all switches in the building block fall into one of three categories (see Chapter 4): unused or faulty, a connecting switch or an image of a switch in the virtual lattice mapped into the building block. The connecting switches do not change configuration settings from phase to phase. Their setting is permanently fixed and serves to provide the reconfiguration necessary to map the virtual lattice into the building block. As a result, connecting switches are always in standby mode. In contrast, the image switches change their setting during a reconfiguration and so must be in active mode.

Each block contains 240 switches. Of these, 21 are image switches. The remaining 219 switches are in standby mode. With a 9 × 9 grid of building blocks on a wafer, there are a total of 19,440 (= 240 × 81) switches on the wafer. 1701 (= 21 × 81) of these are image switches leaving 17,739 switches in standby mode. Now, each switch in the building block is of degree eight so no more than eight memory bits are required to store a switch setting. With four settings per switch (a typical local memory size) and one register to hold the current configuration setting, there are 40 bits of memory per switch.

By scaling down the larger (200 mW) of the cited values for active power consumption for the 64K memory (containing 65,536 bits), we can obtain an approximate upper bound on the power consumed by the local memory of the switches. The total power consumption of the image switches (in active mode) does not exceed

$$\frac{1701 \times 40}{65{,}536} 200\ (\text{mW}) = 208\ (\text{mW})$$

While the image switches are in transition, the connecting switches are idling along consuming no more than

$$\frac{17{,}739 \times 40}{65{,}536} 10\ (\mu\text{W}) = 0.11\ (\text{mW})$$

In total, the switches consume well less than a single watt. Reconfiguring does not tax the power dissipation capabilities of a wafer.

Now consider the power requirements of a CHiP processor in computational mode. The switch lattice connections are fixed so all switches are in standby mode. Total power consumption by the switches is no more than

$$\frac{19{,}440 \times 40}{65{,}536}\ 10\ (\mu W) = 0.11\ (mW)$$

Switch power dissipation is well less than a milliwatt. This is a negligible amount. This leaves approximately the full 3W to be consumed by the processing elements.

It is difficult to estimate the power consumption of a processing element without knowing all its design details and performing detailed simulation studies. So, as with the estimates of the memory power consumption, we will rely on reported power consumption figures for similar devices. In particular, a team at Bell Laboratory designed and fabricated a systolic array processor implemented in twin tub CMOS technology and with several simple PEs per chip. They reported 10 mW / PE power dissipation [West]. Due to the close similarities of the Bell project and the wafer scale CHiP processor, we will adopt a 10 mW estimate for the power consumption of the CHiP processing element.

Processing elements fall into one of three categories: *active* PEs which are images of PEs in the virtual lattice, faulty PEs and PEs which are functional but unused. With four PEs in each virtual lattice and a 9 × 9 grid of building blocks on the wafer, there is a maximum of 324 (= 4 × 81) active PEs per wafer. At 10 mW per active PE, just over 3W are dissipated by the active PEs. With switches consuming a negligible among of power, the target power dissipation of 3W is (approximately) met as long as the faulty and unused PEs consume no power.

Faulty PEs pose no problems. They can be completely disconnected from the lattice and from the power supply by laser trimming or fuse blowing. No power need be consumed by any faulty PE.

On the average, there will be a large number of fully functional but unused PEs. Many of the building blocks will contain more than the minimum number (four) of PEs required to host the virtual lattice. The extra PEs in each block will not be used. Of the 972 (= 12 × 81) PEs on the wafer, approximately 65% are functional. With 324 active PEs, this leaves 972 × 0.65 - 324 ≈ 300 functional but unused PEs. If each of these consumed 10 mW, the total power consumption of the wafer would double. This is unacceptable.

Unlike faulty PEs, it is undesirable to disconnect the functional but unused PEs. Laser trimming (or fuse blowing) physically severes the links to a PE. This is irreversible. Once a PE is disconnected, it can not be reconnected. During the lifetime of the machine, some PEs will undoubtedly fail. We would like to keep the unused PEs in reserve so they can be switched in to take the place of a PE that has failed. If functional but unused PEs are permanently disconnected from the lattice during the initial configuration of a building block into a virtual lattice then the block is left without any redundant PEs. It has no fault tolerance. The failure of a single PE renders the building block faulty which in turn causes the entire column to be excluded. Without fault tolerance, a single faulty transistor within a PE would cause the loss of an entire column of blocks. This is clearly an undisirable situation.

What is required is a *programmable power down* capability. The controller of the CHiP processor must be able to power down a PE so that it consumes negligible power. If in the future, the PE is needed, the controller must be able to power it back up so it can be switched into the virtual lattice. The activation of any PE is programmed by the controller. One mechanism for implementing programmable power down is outlined below. The technique described applies to nMOS technology, but similar methods can be used for PEs implemented in CMOS [West].

Power is drawn when there exists a closed circuit between the power supply and ground. Circuits are opened and closed by transfer gates. If all transfer gates are open ("off"), essentially no power is consumed. With any enhancement mode transfer gate (Figure 5.7.1), the gate is turned on when the potential across the gate and the channel exceeds the threshold voltage, V_t ,of the device. The channel is at the same potential as the substrate, V_{ss}, which is typically kept near zero volts. In other words, the gate turns on when

$$V_g - V_{ss} \geq V_t$$

Normally all PEs are fabricated on the same substrate so all transfer gates have the same channel potential. However, this need not be the case. Each PE can be fabricated in its own separate tub (Figure 5.7.2). The tubs of the different PEs are electrically isolated from one another and hence can be maintained at separate and independent voltages. Setting the value of

$V_{ss} \approx 0$ for a particular PE results in normal operation of the PE. Gates turn on and off for $V_g \approx V_t$. But by raising V_{ss} to a high voltage (*e.g.* $\approx V_{dd}$) prohibits the gate from turning on. $V_g - V_{ss}$ can not exceed the threshold voltage so all gates in the tub remain off. Consequently, the PE is "shut off" and consumes only the negligible amount of power required to maintain the tub at the high potential.

To selectively change the V_{ss} potential of a PE, there must be a selection mechanism controlled by the CHiP controller (Figure 5.7.2). By addressing the PE via the select lines, the controller can choose the tub potential for each individual PE. Note that the switching circuit between V_{ss} and the high / low voltage input must be low impedance. There must be little voltage drop across the switch since this results in reducing the V_{ss} potential. Furthermore, it is necessary that the switch provides a steady and stable V_{ss} potential; it must not pick up noise from surrounding circuitry.

b) Skeleton Routing

There are a number of signals common to all processing elements and switches. The set of wires that must be routed to each and every component on the wafer is termed the *skeleton*. It is the "backbone" of the CHiP processor and provides the power and timing signals plus control and interrupt signals from the CHiP controller. Through the skeleton, the individual components are synchronized, started / stopped and made to function as a harmonious group rather than a cacophony of separate devices.

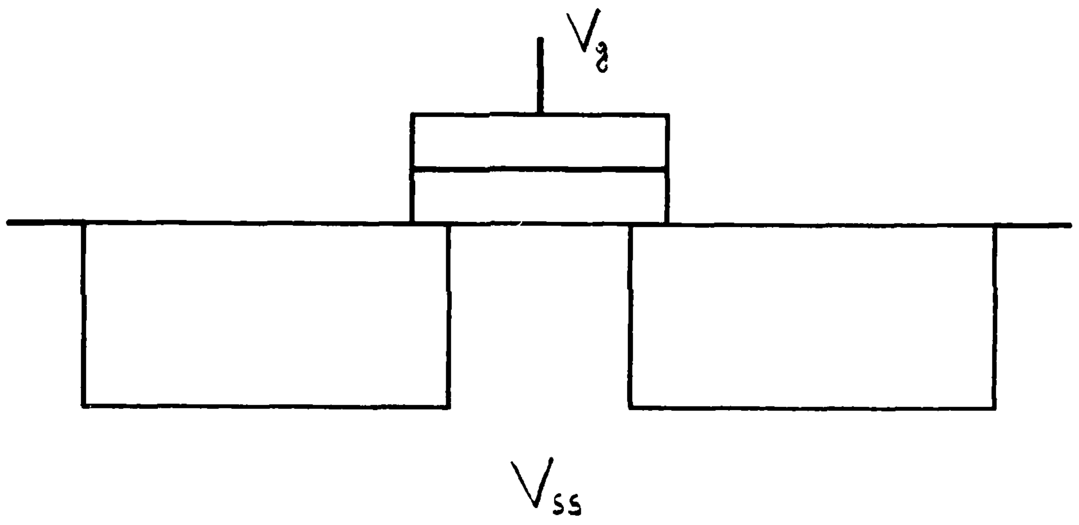


Figure 5.7.1 - Enhancement Mode Transfer Gate with Reference Voltages

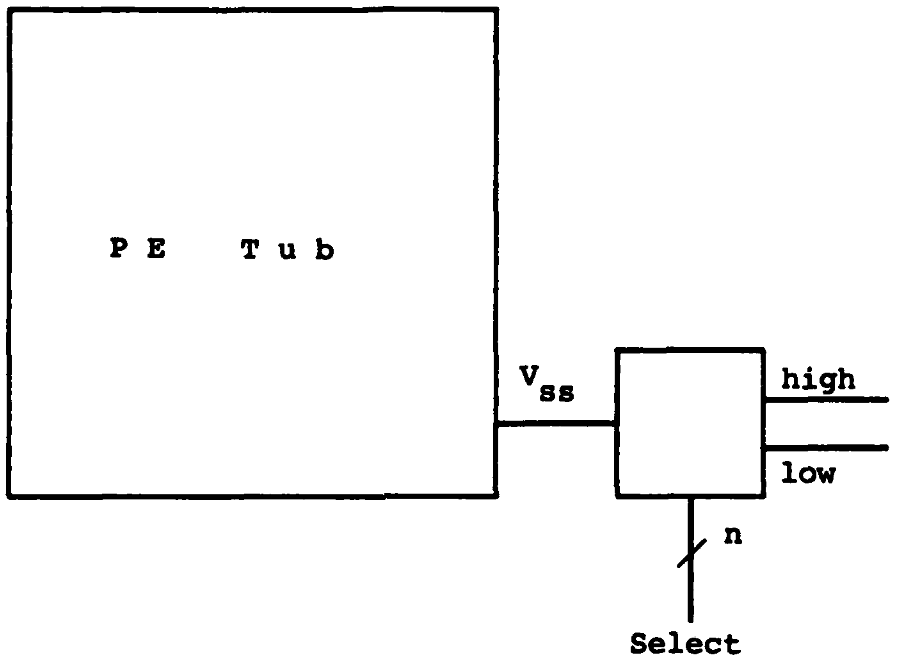


Figure 5.7.2 - Implementation of Programmable
Power Down Mechanism

The skeleton unifies the individual components by sending to every component the same set of control and synchronization signals. This unification is an architectural plus but also an implementation problem. The need to string one set of wires through all components on the wafer introduces problems of wiring integrity, wire routing and signal skew. These will be considered below.

The wiring of the skeleton must be highly reliable. A component with a broken or incomplete connection to the skeleton is faulty. Extremely high wire reliability can be achieved by patterning wide wires with large spacing between them. Making a wire wider than a defect makes it impervious to the occurrence of defects; a defect can not cause a wire beak. Furthermore, increasing the spacing between wires reduces the chances of a short between two adjacent wires. Experience with prototype wafer scale systems at Lincoln Laboratory has shown that metal wires can be run the entire length of a 3" wafer with 100% wire integrity [Chap]. No wire faults have been encountered.

An additional source of difficulty is the topological problem of routing the skeleton connection to all components. The wafer surface is fairly tightly packed with PEs and switches. Between all components runs datapath wires leaving no wiring channels that are completely unused on all wiring levels. Additionally, this problem appears to be exacberated by the extra large width and spacing of the skeleton wires. Even a small number of skeleton wires forms a wide bus.

A solution to this problem is to use the second level of metal interconnection almost exclusively for skeleton routing. Datapaths and switches use only the lower three wiring levels. The PEs can make limited use of the upper level for short, local wires contained within the PE itself. This leaves the uppermost level free for the routing of the skeleton. The wires of the skeleton can run overtop the datapaths, switches and PEs. (In practice, one may need to restrict the overlap of some lines to prevent faults from having global effect. For instance, if an interrupt line from the CHiP controller was shorted to a ground line on a lower level, the entire interrupt line could be disabled.)

The wide wires of the skeleton are well suited to the "rough terrain" found on the uppermost level of interconnection. The design rules for upper metal levels typically require wider wires to reliably traverse the "hills and valleys" left by the patterning of lower levels. The width and spacing of the skeleton wires meets and exceeds these requirements.

c) Clocking

The wires of the skeleton are extremely long compared to all other wires on the wafer. The longest wire between PEs is on the order of the edge length of a PE, less than 2 mm, and the maximum wire length within a PE is of similar length. The skeleton wires must traverse the entire length of the wafer, about 4". The signal propagation time in the skeleton is significantly longer than the signals delays within or between individual components. More importantly, there is a large amount of signal skew - a signal (which originates at a single point on the wafer edge) arrives at different components at different times. Components close to the origin of the signal receive it sooner than those farther away. Furthermore, because of the long propagation time, the amount of skew can be large compared to the cycle time of the PEs. Thus it is impossible to exactly synchronize the components from a common source [Seit79].

This same problem is encountered with programming the lattice. The switch lattice may be configured so that there are long paths of switches between some PEs and short paths between others. This introduces programming difficulties. To maintain synchronous operation, the delay through the longest path must be computed and wait instructions inserted into the programs of the PEs to synchronize the fast and slow communications paths. This further complicates the already difficult problem of programming a parallel machine.

A solution to this timing problem is for the PEs to be locally synchronous but globally asynchronous. Circuits within each PE run synchronously off a common and locally generated clock. However, communication between PEs is asynchronous. PEs must signal to exchange data. No PE is allowed to make assumptions about the timing of other PEs. This asynchrony also applies to the start / interrupt signals from the CHiP controller. Some PEs may start before others, but the signalling protocol forces the early starting PEs to wait for the late starting ones [Cuny82].

CHAPTER 6

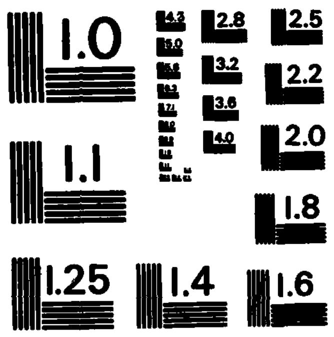
TESTING CHiP PROCESSORS

As systems-on-a-chip become larger and more complex, efficient and complete testing becomes an increasingly difficult problem. The number of possible faults increases and access to internal on-chip test points is increasingly costly and difficult. This difficulty is particularly acute for the current generation of microprocessors. Their complexity of operation combined with their large number of gates poses difficult testing problems. In particular, wafer scale CHiP processors will be an order of magnitude more complex than any other system ever fabricated on a single piece of silicon. This raises the question of their testability.

In this chapter it will be shown that CHiP processors can be efficiently tested. A key to limiting the complexity of the testing problem is modularity. The magnitude of the testing problem can be substantially reduced if a large and complex device can be divided into subsections that can be independently tested. CHiP machines have a highly modular design; they are composed of identical and independent processing elements and switches. In addition, the programmable switch lattice provides flexible and fault tolerant access to the components of the lattice. Furthermore, each of the individual PEs and switches contain only a modest number of gates so

AD-A121 886 WAFER SCALE INTEGRATION OF PARALLEL PROCESSORS(U) 3/3
PURDUE UNIV LAFAYETTE IN DEPT OF COMPUTER SCIENCES
K S HEDLUND NOV 82 CSD-TR-411 N00014-80-K-0816
UNCLASSIFIED F/G 9/2 NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

designing switches and PEs to be easily testable is a manageable problem. These factors contribute to the testability of CHiP machines.

Although CHiP machines are testable (a claim that will be proved later in this chapter), the problem is non-trivial. First, a complete CHiP machine is a very large collection of gates and transistors. Modularity breaks this complexity into smaller pieces, but still the number of gates to test is enormous. As shown in the previous chapters, on the order of 1000 processing elements and 20,000 switches can be fabricated on a single wafer. The size and scale of the testing problem is formidable.

Secondly, the simplest way to test a collection of objects is to individually test each object. But, in a CHiP machine, the only access to the lattice is through the switches on the lattice edge. No other components can be directly reached. In particular, the components in the center of the lattice can be reached only by traversing a long path of switches. The mesh structure of CHiP machines allows direct communication only from component to component, This does not facilitate testing.

To increase accessibility, it is possible to put bonding pads at each component. An external testing device can probe the pads to individually test each component. This is called *probe based* testing. PEs are much larger than bonding pads so their addition increases the PE area by only a small amount.

However, switches are quite simple and small. Typical switch dimensions are 250 × 250 μm (in $\lambda = 1$ μm technology). Placing bonding pads around the periphery of the switch introduces approximately thirteen access points and more than quadruples its area. As the size of transistors continues to decrease, switches will become smaller but the area of a bonding pad will not decrease. Thus the overhead due to the bonding pads will increase with progress in semiconductor technology. Additionally, the presence of bonding pads around the periphery of the switch causes severe routing problems for the datapaths. As a result of these difficulties, it is impractical to put bonding pads for testing at the switches. Switches can not be directly accessed by a testing device. They must be indirectly tested. *Indirect testing* means a component is accessed only by going through a path of intervening switch (see Figure 6.0.1).

The testing process requires two actors: the device being tested and the testing device that controls and performs the tests. With indirect testing, there must be a path of switches with a testing device on one end and the device being tested on the other end. The switch path connects the two actors and establishes a communication path over which testing is performed. However, switches are *passive*. They serve only to connect two incident datapaths. As a resu'., the actors can not be switches. Actors must be either PEs or an external testing device.

In general, some of the switches along a test path switches will also be untested. Any of the untested components on the path may be faulty. As a result, if a test fails, the cause of the failure is not pinpointed by a single test. As an additional complication, more than one component on the path may be faulty. To narrow down the possible sources of failure, additional tests on different but intersecting paths must be performed. However, these paths may in turn introduce new untested components. If one of these new tests fails, additional paths must be tried, and so on. Selecting sequences of

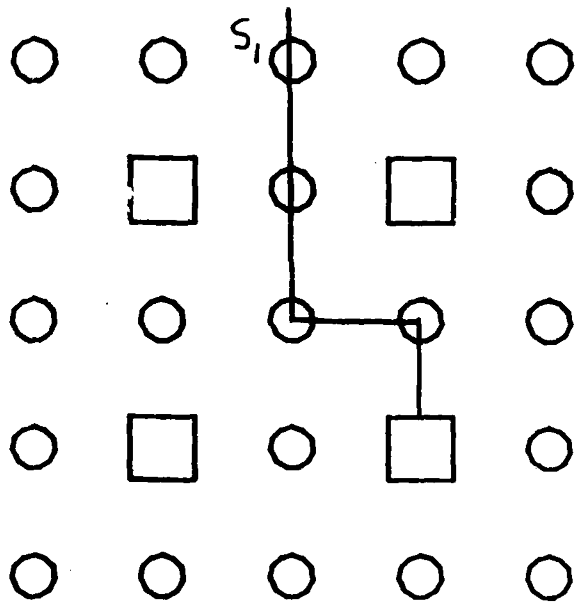


Figure 6.0.1 - Indirect Testing via a Path of Switches

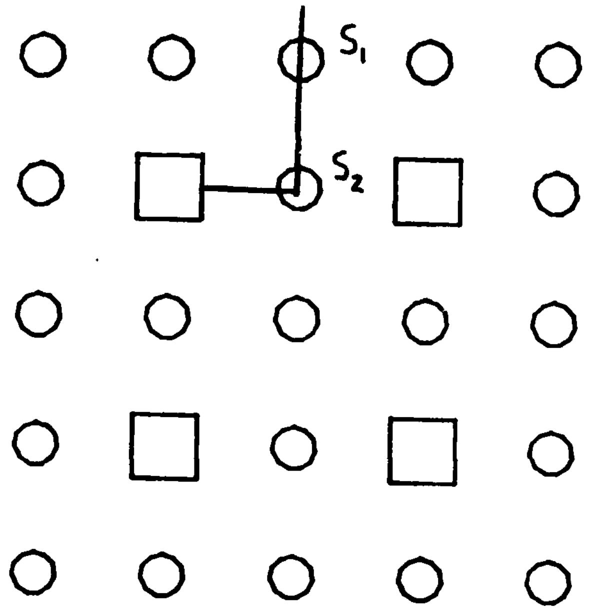


Figure 6.0.2 - Testing with the Reflective Switch

test paths to deduce the location of faulty components is non-trivial.

In fact, indirect testing in the worst case requires performing an exponential number of tests in order to competely test the lattice. For example, suppose that there is only one functional path in the lattice. All other switches and PEs not on the indicated path are faulty. Therefore any test performed on any path other than the single functional path will fail. The only way of testing any of the components on the functional path is to attempt a test along the good path. Since there are an exponential number of different paths in the lattice and the single good path may be randomly chosen, exhaustive search is the only means of locating the functional path.

The cause of this exponential explosion is that it is impossible to incrementally test the components of the lattice. Ideally, we would like to be able to test one component at a time. Starting by testing (with an external testing device) one switch at the lattice edge, testing would fan outward from this one point. In each step, all neighboring components are tested.

The passive switch prevents incremental testing. Since a switch can not be an actor, a PE must be at one end of the test path. This requires multiple components in the path so that a test along the path does not test a single component but several components simultaneously. If the test succeeds then all components along the path are known to be good. However, difficulties arise when the test fails - we do not know which component in the path is faulty. There may even be several bad components. Exponential explosion results from this lack of information that can be inferred from an unsuccessful test. A solution presented below is to redesign the switches to be active.

What is required in order to be able to incrementally test switches? A switch is tested by specifying its configuration setting and then verifying that the switch can successfully transmit a preselected set of bit patterns. This set includes patterns that check that no data lines are open, shorted or capacitively coupled. For example (see Figure 6.0.2), the NS setting of switch S_1 is tested by a testing device (TD) to its North. The testing device sends each signal of the set through switch S_1, waits for the actor on the other end of the test path to return the signal and verifies that the returned signal matches the original one. This sequence is repeated for each pattern in the set. Thus if switch S_2 could reflect back a signal coming from the North, S could be tested in isolation. This is called the ***reflective switch***.

One control line in the datapath is used to put switches into reflect mode. Note that to correctly reflect the signal, the switch must latch the incoming data, wait for the datapath lines to become quiescent and then return the latched data.

With every switch in the lattice being a reflective switch, it is possible to step through the lattice testing one switch at a time. The testing problem is reduced to a linear time algorithm. For example (see Figure 6.0.2), the following steps incrementally test the components on the path from the testing device to the East port of the PE:

1) S_1 reflect North - the datapath between TD and S_1 is tested.

2) S_1 set to NS and S_2 reflects North - test the NS setting of S_1.

3) S_1 set to NS and S_2 to NW - this completes the connection to the East port of the PE testing the NW setting of S_2.

A longer path of switches can be tested simply by repeating step two.

To implement the reflection mechanism, each switch has added to it circuitry used solely for testing. Each switch has two components: its switching mechanism and its reflection circuitry. This reflection circuitry is not independently tested. A switch is considered good only if both the switching mechanism and the reflection circuitry are functional. Thus some switches will be lost due to faults in the testing circuitry even when the switching mechanism is functional. But since the reflection mechanism is relatively simple, faults in it should be infrequent. On the other hand, this mutually coupling between the two components of the switch greatly simplifies and speeds the testing process. The loss of a few switches is an acceptable price to pay.

Note that if S_2 due to internal faults can not reflect back a signal (Figure 6.0.2), the test of the NS setting of S_1 will fail. The internal switching mechanism of S_1 may be functional but if the device to which it will communicate if faulty then so is S_1. All three settings of S_1 that connect to S_2 (NS, SE and SW) are termed *connectivity* faults.

To recapitulate, there are two approaches to lattice testing. Both require indirect testing. One uses passive switches, and in the worst cast requires and exponential number of tests to completely test the lattice. The other method employs a reflective switch. The added switch complexity causes a slight reduction in switch yield, but allows incremental testing of components. The entire lattice can be tested in linear time. Both approaches have their applications.

Note that the example of worst case exponential behavior with passive switches necessitated a large number of faulty components. There was only a single functional path in the lattice. But in practice, switches will have high yield. Most tests will succeed, not fail, so exponential explosion will be avoided. Furthermore, it is desirable to keep switches simple (see Chapter 4). This enhances their yield since there are fewer devices to fail in a simple switch than in a complex one. Additionally, the simpler switches occupy less area leaving more room on a chip for PEs.

As a result, the passive switch approach is desirable for small lattices. For example, section 2.5 considers embedding a 2×2 lattice within a 3×2. Lattices of this size are small enough to allow even exponential testing (although it should be extremely rare). The benefits derived from the passive switch outway the need for a reduced bound on testing time.

On the other hand, a wafer scale CHiP processor presents a testing problem of far greater magnitude. With over 20,000 components per wafer, efficient testing is a necessity not a luxury. Reflective switches are required. This insures linear time testing at a cost of some switch yield and switch area. The advantages of wafer scale integration are not without their costs.

In the following sections, we will briefly explore the design of switches and PEs for testability and investigate algorithms for testing a lattice with passive switches.

1. Design for Testability

A primary restriction on CHiP processors is that the number of connections between PEs should be kept to a minimum. The area of a switch is proportional to the square of the number of wires in the datapath. To avoid the area of a CHiP lattice being dominated by inter PE wiring, the number of connections between PEs must be limited. Since testing is performed through the switch lattice, we need to be able to test PEs with using only a small number (*e.g.* 8-16) of connections between the PE being tested and the testing device.

However, this requirement is not unique. Designers of conventional single chip microprocessor and other LSI devices need to devote as few of the scarce bonding pads to testing as possible. This is the same requirement as for CHiP machines; the number of connections between the processor and the testing device must be minimized through use of these techniques.

As a result, standard testing techniques and design for testability principles developed for conventional microprocessors [DasG78, Haye80, Will73, Will79] carry over to the testing of the PEs of CHiP processors. Use of scan in / scan out shift registers, signature analysis, etc. can be incorporated into the design of PEs to compress the output of the testing process. The number of datapath lines required to carry this information is minimized.

Furthermore, the PEs of a CHiP machine are quite simple in comparison to the current generation of advanced microprocessors (see Chapter 4). Therefore the problem of generating test sequences to throughly exercise a PE is a comparatively simple (although certainly non-trivial) problem. Consequently, it will be assumed that the problem of testing individual PEs can be solved by conventional techniques. Additionally, switches are considerable simpler than PEs. Their testing poses no difficulties.

2. Model of Lattice Testing

In this section an abstract model of lattice testing with passive switches is presented. This model formalizes the mechanics of the testing process so that proofs of testability can be presented. Furthermore, the essential aspects of testing are identified and isolated so that the testing process can be examined without being mired in detail.

Due to the modularity of CHiP processors and the independence of the components, testing splits into two separate problems:

- testing individual components
- testing the lattice as a whole.

In the remainder of this chapter, we concentrate only on the second problem, that of testing the lattice. We assume there are well-defined sequences of testing steps to thoroughly test the individual processing elements and switches. The specific sequences depend intimately on the design details of the components; this will not be considered further.

We present a model of the problem of testing lattices of PEs and switches. The model specifies:

- components of the lattice which must be tested
- requirements for a complete test
- goals of the testing procedure

The model is at a high level of abstraction. It does not deal with responses to specific test patterns, the mechanisms of performing the testing, or details of generating the test data. These factors will vary greatly with changes in the implementation details of a specific CHiP machine. The resulting model achieves independence from the myriad of design details underlying the overall machine architecture. It captures the essential problems of testing complex lattices of PEs and switches without being tied down to specific implementations of the components. This allows formal descriptions of testing algorithms without excessive and obfuscating detail.

a) Definitions

In this section certain key concepts concerning testing and the lattice structure are precisely defined. This replaces intuitive notions of testing and testability with sharply defined and delimited concepts. Through this approach, the fault coverage of a testing procedure can be formally determined, and the correctness of a testing algorithm can be proven.

There are two actors in the testing process:

a) Processing element being tested, also referred to as the unit under test (UUT).

b) Testing device (TD). This controls the UUT, applies test signal to the UUT, monitors the response and is responsible for deciding if the UUT is functional or faulty.

The testing device may be external to the lattice - a separate and independent device. It may be special purpose testing equipment such as a programmable logic analyzer or a general purpose computing device such as the CHiP controller. An external device can access the component being tested directly by probing the bonding pads of the UUT. Indirect access is also possible. A subset of the switches on the lattice edge, ***gateways*** , are connected to bonding pads. The external device can access the UUT via a path of switches originating at a gateway.

In addition, the testing device can be another PE in the lattice. In this case, a small subset of the PEs are initially tested by an external testing device. The PEs found to be functional are used to test their neighbors which in turn test other PEs, etc. This is a self testing strategy which is initiated by a limited amount of external testing.

A single ***testing step*** consists of three distinct phases:

1) Generation of test data - the input test pattern to the UUT and the correct response.

2) Application of the input test pattern to the UUT.

3) Evaluation of the response. This most commonly consists of comparing the response to the known, correct value. Other characterizations of the response such as number of 1's (bit count) and number of transitions from 0 to 1 (transition count) can also be used.

A testing step is an exchange of signals between the testing device and the UUT. The TD initiates the testing step by presenting an input pattern to the UUT which is under the control of the testing device. In addition to data, the input pattern may include instructions for the UUT to execute. Thus a typical testing step starts with the TD downline loading the UUT with a small program segment and input data. The UUT executes the code while the TD monitors the output and halts the UUT at the completion of the testing step.

An individual testing step can verify that the UUT correctly executes a single program segment. A *test* of a component is a sequence of testing steps which thoroughly exercises the component and provides adequate fault coverage. A *test is successful* only if every testing step succeeds.

Some basic lattice terminology will be introduced. Processing elements have a port at each compass point, NSEW, through which the PE can communicate with its neighbors. Each switch is also connected to its four neighbors. A *configuration setting* specifies which pair of incident datapaths the switch will connect. There are six possible switch settings (NW, NE, SW, SE, NS, EW). Each setting is denoted by the pair of compass points that are connected. Lattice elements are matrix numbered with zero origin. A *path* through the lattice is a connected sequence of switch settings with, optionally, a port on either end. The *components* of the path are the individual switch settings and ports. When the specific switch settings are unimportant, a "generic" path as in Figure 6.2.1 will be specified where the setting of switch S is assumed to be the one required to connect path segment P to $PE[i,j]_E$.

b) Testable Components

Datapaths are not explicitly tested but rather are tested in conjunction with switch testing. A fault in a datapath is reflected by faults in the

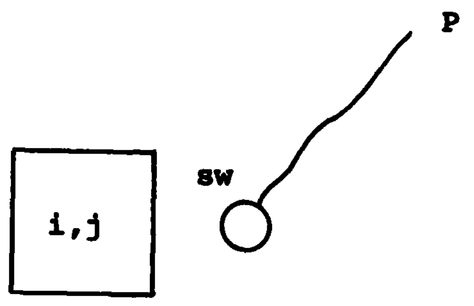


Figure 6.2.1 - Example of a Generic Path

components connected to the datapath. For example the datapath fault results in faults in PE_E and the NW, EW and SW settings of switch SW.

An intrinsic fault is caused by a defect within the lattice element which causes erroneous behavior. Broken wires or shorted transistors are examples. Any component incident upon an intrinsic fault is also faulty. If the East port of a PE is faulty, so is the West side of the adjacent switch. Settings S_{NW}, S_{EW} and S_{SE} are termed *connectivity faults* since they are attached to an intrinsic fault.

Each of the six switch settings are considered independent and can be individually good or bad. Analogously, ports are independent. For any given PE, some of its ports can be functional and others faulty.

Both switches and PEs have internal mechanisms in addition to their observable communication behavior. A switch must be able to latch new settings sent to it and select amongst those settings stored in its local memory. A PE consists of a processor which interprets the PE's instruction set and four ports. A PE must correctly execute its full instruction set and have an fault free memory. A failure in the internal mechanism of a switch or the processor of a PE causes all its settings or ports to be considered faulty. Each individual component is good only if the internal mechanism is fully functional. There is no point to communicating with a faulty PE nor using a switch which can not reliably select its setting.

Testing the internal mechanism of PEs and switches will be implicitly assumed. When "test West port" is specified in a testing algorithm, it is assumed that the first port of a PE that is tested also includes a full test of the internal mechanism of the PE; similarly for switches. As a result, we can be concerned with only testing ports and switch settings.

Switches are not directly accessible from testing devices. A switch setting is tested by establishing a path between two PEs (or external testing devices) and performing the sequence of testing steps required to fully exercise the switch and the datapath. In general, a path may contain more than one untested switch setting. Consequently, a failure of the test along a path will not necessarily pinpoint the faulty component. In fact, there may be more than one defective device on the path. Hence, a test can, in general, verify the functionality of components but an unsuccessful test required that tests along additional paths be performed to locate the fault(s). In summary,

a *switch setting is functional* if it is on the path of a successful test

a port can communicate if it is on the terminating end of a path which is successfully tested. A *port is functional* if it can communicate and the internal mechanism of the PE functions correctly. To conclude that a port can not communicate, it must be impossible to successfully test the port via all three access routes into the port (see Figure 6.2.2). If a test along any one of these access routes is good then the port is good.

In general, an unsuccessful test along a path with more than one untested component does not provide any new information on the status of the untested components. Any combination of untested components of the path may be faulty. A single test does not separate the possible combinations of faults. One important example is:

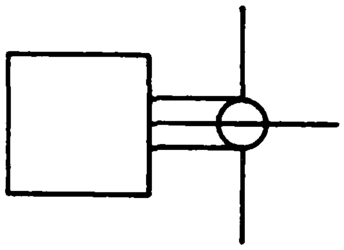


Figure 6.2.2 - Three Paths Required to Test a Port

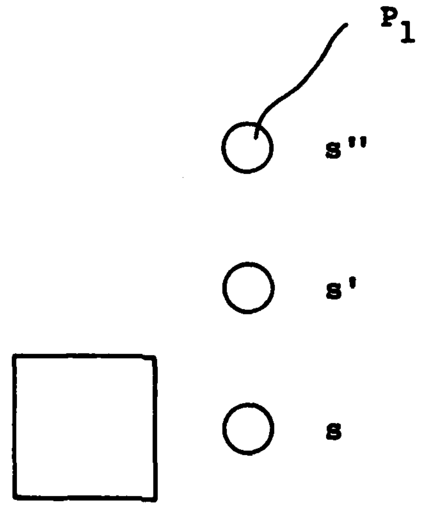


Figure 6.2.3 - Testing a Port

Lemma 2.1a - Given the path in Figure 6.2.3 with path segment P_1 S'' S'_{NS} good and both S_{NW}, $PE1_E$ untested, the status of S_{NW} is determined by the test along the path

$P = P_1$ S'' S'_{NS} S_{NW} $PE1_E$

independently of the status of $PE1_E$.

Proof -

Case 1 - $PE1_E$ is good.

If S_{NW} is good then all components of path P are good and the test along P succeeds. Otherwise the test fails.

Case 2 - $PE1_E$ is bad.

The test along path P will fail. This is correct since S_{NW} is a connectivity fault.

QED

The above lemma is easily generalized to

a) any port of the PE

b) allowing S' to occupy any position adjacent to S and S'' any position adjacent to S'

This generalization is stated somewhat informally:

Lemma 2.1 - Any switch directly connected to a port can be tested independently of the status of the port if there exists a good path from a gateway to the switch.

c) Goals of a Testing Procedure

In a CHiP machine, every component must be fully operational. However, the switches and PEs fabricated on the wafer may be only partially functional. In a PE, the processor may work but one of the ports may be dysfunctional. Also a switch may have only a (proper) subset of its settings working correctly.

Partially functional components may serve a useful function in a fault tolerant machines although they will not be an integral part of the virtual CHiP lattice. A partially functional switch may serve as a connective switch providing a path between two fully functional components. Additionally, a PE with at least two good ports may be used in the self testing of the lattice.

As a result, a go/no-go test for PEs and switches is insufficient. The goal of any testing procedure is to provide fault location at the component level. It is necessary to know which settings of every switch and ports of every PE are good even though the device may only be partially functional.

Below the component level, fault detection is sufficient. For example, if a switch setting is bad, it is not necessary to know which particular transistor(s) are defective. If the processor of a PE is faulty, knowing whether the memory, datapath or control logic is the culprit is unimportant.

Furthermore, the testing algorithm must provide complete component-level resolution. It is unacceptable to have otherwise functional components reported as faulty due to limitations of the testing algorithm in resolving the source of errors.

In addition to providing reliable, component-level fault location, any testing algorithm must be efficient. A wafer scale CHiP machine is a very large collection of components. A processor fabricated on a 4" wafer consists of over 20,000 switches and 900 PEs. An inefficient testing algorithm will be computationally intractable.

3. Lattice Testing

Given an arbitrary port in the lattice, what are the requirements for testing it? First, the port must be connected to a testing device. An external testing device can access the port via a functional path from a gateway to the port. The port may also be tested by another PE in the lattice. But this PE doing the testing must be previously tested. So the testing PE must have a functional path to a gateway or to another PE which in turn has a path to a gateway or ... As a result, only regions of the lattice which are connected to a gateway can be tested but with the connecting path allowed to pass through intervening PEs. If a component is not accessible from a gateway, it is untestable and hence considered faulty.

A region may be functional but if it is disconnected from the remainder of the lattice, there is no way to use the region; it can not communicate with the other PEs. So this testing assumption that inaccessible regions are faulty does not cause the loss of otherwise usable PEs.

Secondly, it would be ideal if all the components on the path to the gateway were known to be functional. A successful test verifies the functionality of all components on the path. But an unsuccessful test, in general, fails to pinpoint the source of the failure. The fault can be located by a single test only if there is exactly one untested component on the path. Otherwise, additional tests (perhaps a large number) are required to isolate the defective component.

However, testing a port and testing the switch to which it is connected can not be separated. A port can not be accessed independently of its switch. Similarly, the West side of switch S can be tested only by being on a path that terminates at the PE. The switch and the port are mutually coupled with respect to testing. They must be simultaneously tested.

Because of this coupling, the primitive unit that will be tested is a port pair, two adjacent ports and the intervening switch (Figure 6.3.1). A single port and its switch could have been chosen but, as will be seen, testing can proceed by pairs of ports as easily as individual ports.

What are the requirements to be able to test a port pair? To test ports $PE1_E$ and $PE2_W$ through S' (see Figure 6.3.1), there must be a functional path from S' to a gateway, and S must complete the connection from S' to each port. When these two conditions are met, we say that S' is a hook since it allows the testing device to latch onto the port pair for testing. In the worst case (*e.g.* a faulty port), testing a port requires that all three access routes into the port be attempted. So, both S' and S'' must be hooks for the port pair. We say S' and S'' are a ***hook pair*** for the port pair. Furthermore, since both the North and South sides of switch S must be accessible from a gateway in order to fully test the ports, the existence of ***a hook pair is the minimum requirement for completely testing a port pair.***

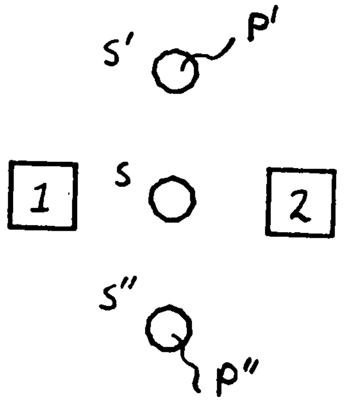


Figure 6.3.1 - Testing a Port Pair

The following algorithm can be used to test a port pair.

Port Pair Test Algorithm

Input - a port pair (see Figure 6.3.1) with a hook pair S' - S" and test paths to a gateway P_1 and P_2.

Output - status of all components in the port pair.

Remarks - Initially, all components are marked FAULTY. If a test succeeds, all components on the test path are marked GOOD.

Mark all components FAULTY.

T_1: test S_{NS} via path P' S' S_{NS} S" P"

(The following paths all use the segments P' S' or P" S". They will be omitted for clarity.)

T_2: test $PE1_E$ via S_{NW}

T_3: test $PE1_E$ via S_{SW}

T_4: test $PE2_W$ via S_{NE}

T_5: test $PE2_W$ via S_{SE}

T_6: test along path $PE1_E$ S_{EW} $PE2_W$

Test T_1 exercises the NS setting of switch S which is connected to the testing device via path P' and switch S' and path P" and switch S". The four test paths in T_2 - T_5 have only one connection to the testing device. The other termination of the path is a port.

After the completion of tests T_1 - T_5, the only untested component is the EW setting of switch S. This test is qualitatively different from the others since it is by necessity a self test; the two ports must test the setting themselves. Self testing is possible only if both ports are functional. The code for test T_6 is downline loaded into each PE via one of the functional paths found in tests $T_2 - T_5$, test T_6 is performed and the PEs report the results back to the testing device.

Theorem - If S' and S'' are a hook pair for R (with test paths P' and P'' respectively, see Figure 6.3.1) then the PE Pair algorithm tests $PE1_E$, $PE2_W$ and all settings of S, despite faults.

Proof - S_{NS} is tested since the path P' S' S_{NS} S'' P'' contains only good components except for S_{NS}. No other components in R affect this path so faults in other components will not alter the testability of S_{NS}.

By Lemma 1, S_{NW}, S_{SW}, S_{NE} and S_{SE} are tested regardless of the status of the incident port. No other components in R can affect the test paths used to test these settings so S_{NW}, S_{SW}, S_{NE} and S_{SE} are tested despite faults in the port pair.

We must show that $PE1_E$, S_{EW} and $PE2_W$ can be tested despite faults in R.

Consider the situation immediately before T_6 in the algorithm, and let P be the path $PE1_E$ S_{EW} $PE2_W$.

Case 1 - in the previous testing steps $T_1 - T_5$, we found both ports to be good. We need only test S_{EW}. Path P tests S_{EW} since the other two components in the path are good.

Case 2 - One port is good and the other could not be accessed by either of the paths attempted. Assume $PE1_E$ is the port known to be good. Now, path P tests S_{EW} and $PE2_W$ simultaneously since we have tried both other access routes into $PE2_W$ (*i.e.* S_{NW} and S_{SW}). If this one fails then $PE2_W$ is faulty and S_{EW} is a connectivity fault.

Case 3 - Neither port has been accessed by either of the paths attempted. A test along P tests all three components simultaneously. S_{EW} is good only if both $PE1_E$ and $PE2_W$ are good. This the last access route into either PE so this is the last chance to be able to communicate with either port. Either all three components are good or all three are bad.

QED

Now consider testing the entire region surrounding a PE - a ***PE square*** (see Figure 6.3.2). The "internal" settings of the square are tested by combining four port pair tests as in Figure 6.3.5. Thus forming a *cross* test.

Theorem - If each pair of corner switches, C1 - C4 (see Figure 6.3.6), is a hook pair for the intervening port pair and $P_1 - P_4$ are functional paths to the gateway for the corresponding corner switch in which

a) do not intersect the PE square

b) do not pairwise intersect

$P_1 \cap P_2 = P_2 \cap P_3 = P_3 \cap P_4 = P_4 \cap P_1 = \phi$

then the internal settings of a PE square can be completely tested.

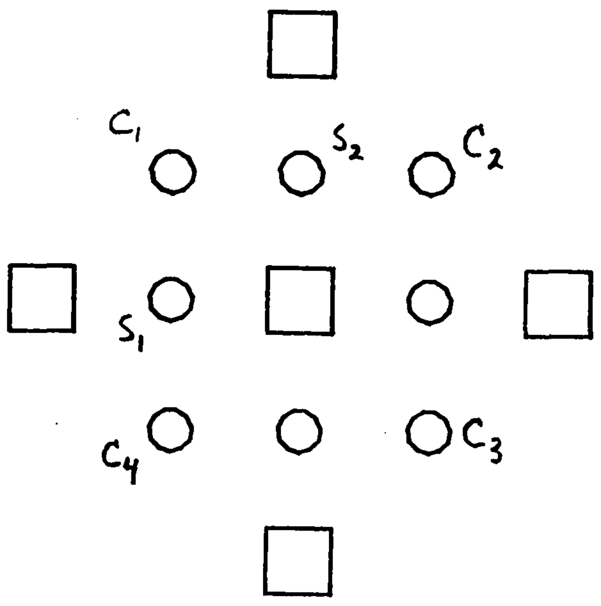


Figure 6.3.2 - Testing a PE Square

Proof - Apply the port pair test algorithm to the four port pairs. By the previous theorem and the assumption that there are hook pairs for each port pair, this tests all four ports of the PE and the four associated switches. All that remains to test is the inside settings of the four corner switches C1 - C4. By symmetry, we need consider only one of these. Choose $C1_{SE}$. If there are functional test paths from both the West side of switch S2 and the North side of S1, $C1_{SE}$ can be tested via these paths. If neither switch has a test path, $C1_{SE}$ is untestable and hence faulty. Finally, assume there is no test path from one switch. Let it be S1. As a result, it is impossible to test any setting of C1 incident upon the North side of S1. So $C1_{SE}$ is faulty.

How are the test paths from the switches found? Consider switch S2. If there is a good connection from S2 to C2, path P2 suffices. Otherwise one of the PEs to the North or South of switch S2 must be the terminating point of a path. If neither of these are functional then the path from C1 runs into a dead end and so in untestable and faulty. Similarly for S1. There are three possible paths from each of S1 and S2 so at most nine paths need be tested.

QED

Note that when PE square tests of adjacent squares are composed, the untested switch settings are precisely those that are tested by the adjacent PE squares. The "external" switch settings of a PE square are precisely the "internal" settings of a neighboring square. Consequently, the cross tests can be composed leaving only the setting on the outer edge untested. But it is the outermost edge of switches which is accessible to the external testing equipment. Thus the entire lattice can be tested.

Theorem - Given any lattice, if all the corner switches are hook pairs for the four neighboring PE pairs with non-intersecting test paths to a gateway, the lattice can be completely tested, despite faults.

Proof - Consider a single square. By the above theorem, the square is completely tested (despite faults) except for the corner switches. Consider the four neighboring squares which form a 5 by 5 lattice. Perform cross tests independently at each of the four squares. The corner square S at the center of the lattice has all right angle settings tested since it is a member of all four squares. We must test S_{NS}, S_{EW}. Consider SW_{NS}. If there are test paths (non-intersecting) from S_N and SW_S then S_{NS} can obviously be tested. Otherwise, there is no test patn from at least one direction North or South. Let it be North. Then S_N is dead by the definition of the testability of a switch and S_{NS} is a connectivity fault.

Similarly for S_{EW}. Hence SW can be completely tested. Consequently, when composing groups of four squares, all components are completely tested except for the corner switches on the edge of the region.

We could similarly show that composing four of the 5 by 5 regions yield a 9 by 9 region with all components completely tested except for the corner switches on the edge of the region. By induction, we can show this holds for any 4n+1 by 4n+1 lattice segment.

Clearly, the corner switches on the edge of a chip can be tested by the external testing device and the neighboring switches (which are already completely tested by the cross tests). Hence, any 4n+1 by 4n+1 lattice is completely testable.

If a lattice is of dimension $m < 4n_1+1$ for some n_1, it is clear that it can be tested in the same manner we would test a 4n+1 by 4n+1 lattice but with the external testing device filling in for the PEs of the larger lattice which fall outside the boundaries of the smaller m by m lattice.

Conclusion - a lattice of any size can be tested.

QED

So far, we have shown that if we have hook pairs then the lattice can be tested. How do we determine that S' and S" are a hook pair? Just as testing ports and their adjacent switches are mutually coupled, so are checking for a hook pair and testing the port pair. The existence of functional paths P' and P" can be determined independently of the status of the port pair. However, the connection from S' and S" to S must obviously involve S. Additionally, completing the connection from S to the ports required that all components of the port pair are involved in the hook pair test. If portions of the port pair are faulty, we may not know whether or not we have a hook pair. This makes it impossible to know if the fault is at the S' - S or S" - S connection or within the port pair. In conclusion, testing for a hook pair and testing the port pair are inseparable.

Algorithm - locating a Hook Pair

Input - a port pair with P' and P" candidate paths to the lattice edge (see Figure 6.3.1).

Output - S' – S" a hook pair? YES/NO returned.

T_1: test along the path P' S' $S_{N,3}$ S" P"

C_1: if successful then YES

Otherwise,

T_2: test along the path P' S' S_{UL} PE_E

T_3: test along the path P' S'S_{UR} PE_W

C_2: if neither T_2 nor T_3 succeed then NO

T_4: test along the path P'' S'' S_{LL} PE_E

T_5: test along the path P'' S'' S_{LR} PE_W

C_3: if T_4 or T_5 succeed then YES else NO

Theorem - Given a port pair with candidate test paths P' and P'' which do not intersect, the Hook Pair algorithm is a decision procedure for the predicate

Q = (P' good) & (S' and S'' are a hook pair for R) & (P'' good)

Proof - A. We must show that if the algorithm returns YES then Q is true. Consider statement C_1 of the algorithm. If T_1 is successful then we know P' and P'' are good and we have verified that both S' and S'' have a good setting which connects the test path (P' or P'') to SW. Consequently, S' and S'' are hooks for Q. By definition P' and P'' do not intersect so S' and S'' are a hook pair for Q.

Consider statement C_2. If either T_2 or T_3 succeed then we know P' is good, and we have verified that the setting of S' connecting P' and SW is good. Consequently, S' is a hook for Q.

Similarly for T_4 and T_5.

If we reach statement C_3 and either T_4 or T_5 succeeds then both S' and

S" are hooks for Q. Since P' and P" do not intersect, S' and S" are a hook pair for Q.

B. We must show that if Q is true then the algorithm returns YES.

Assume Q is true. We then know P' and P" are good and S' and S" are each hooks for Q. Consider S'. There must be a good setting of SW which completes a path to either S" or a good port. There are three settings of SW incident upon S'_3. The algorithm attempts paths with all three so it will locate the complete path and one of the tests T_1, T_2 or T_3 will succeed. Similarly for S" so either T_3, T_4 or T_5 will succeed. Consequently, the algorithm must terminate at either C_1 or C_3. Both of these statements report YES.

QED

What have we accomplished so far? We have reduced the problem of testing the lattice in the presence of faults to locating pairs of hooks. The above theorem reduces this problem to finding pairs of non-intersecting test paths.

test lattice < locate hook pairs < locate test paths

The first reduction is not strictly true since we have considered only the subproblem of testing the lattice when all corner switches are hook pairs for the neighboring PE pairs. Testing a square with an incomplete set of hook pairs wil be considered in a separate section of this paper.

We next examine the problem of locating all possible test paths from a given lattice element.

Theorem - given a lattice element, there are only a finite number of candidate test paths from the element.

Proof Outline - Paths do not have cycles.

At each lattice element along a path, there are only 3 choices for the successor.

The number of lattice elements is finite.

=> the number of possible test paths < 3 ** (number of lattice elements)

QED

In addition to being finite, the set of all candidate test paths from a given lattice element can be listed.

Algorithm - Enumerate all candidate test paths

Outline of Method - Tree Traversal Algorithm

At each component along a path, there are three possibilities for its successor. Faulty components or components already on the path are not legal successors. A path terminates at any port or a switch on the lattice edge.

The key to *efficient* testing algorithms is quickly enumerating candidate test paths. This can be done by:

1. Testing from the edges of the lattice inward.

2. Limiting the maximum length of a test path.

We will show algorithms for testing without considering their efficiency.

The Hook Pair test applies to a given pair of test paths. If we enumerate all *good* test paths from SW' to SW" and apply the Hook Pair algorithm to all pairs of good test paths, we can determine if SW' and SW" are a hook pair for R.

Algorithm - *Complete* PE Pair test

Given a set of good test paths from SW' (S_1) and SW" (S_2) not intersecting R,

```
for every path in $S_1$ do
  for every path in $S_2$ do
      if the paths do not intersect each other then execute Hook Pair alg
      if algorithm  returns YES then
          S' and S" are a hook  pair for R
          test R by PE Pair alg
          return TESTED
          STOP
```

CHAPTER 7

CONCLUSIONS

1. Summary of Results

The key problem in the implementation of wafer scale integration is structuring the wafer so that only the functional PEs are connected together. A methodology, the two level hierarchy, that efficiently and economically solves the structuring problem for CHiP processors has been presented. The principle elements are the use of column exclusion with high yield building blocks that contain redundant components. This approach limits the performance degradation due to structuring and allows the structuring problem to be solved with tractable computational effort.

Since the yield of building blocks must be high for the two level hierarchy to be a practical approach, yield phenomena were investigated in detail. A model of the integrated circuit manufacturing process was developed that predicts circuit yield and the probability distribution of manufacturing defects. These results were applied to the analysis of parallel processors in which several PEs occupy a single chip. In addition, they were used to design the building blocks meeting the requirements of the column exclusion strategy.

It was shown that these building blocks can be assembled into a wafer scale CHiP processor. With current technology, it is possible to fabricate a wafer scale system with 250 to 300 PEs. This represents a truly large parallel machine. Furthermore, this machine is highly robust to faults occurring during the machine's lifetime, consumes a manageable amount of power and can be efficiently tested.

Although the techniques for implementing wafer scale integration were developed for CHiP processors, they can be applied to other system composed of uniform parts. This generalization is discussed in the following section. Furthermore, building blocks are useful on their own; they need not be assembled into a wafer scale system. A generalization of the design methodology used for building blocks is shown (section 3) to increase the maximum allowable chip area and thus increase the number of components per chip.

2. Implementation of General Wafer Scale Integration

The techniques described above for implementing wafer scale integration are not restricted to CHiP processors. The methodology benefits from the fact that the mechanism needed for structuring, the switch lattice, is an integral part of the CHiP architecture. Although this simplifies the work, it is not necessary. The method is entirely general. It can be applied to other systems composed of uniform parts.

As long as a system can be subdivided into modular and independent parts, the switch lattice can provide the flexible interconnection network required to route around faulty components. The settings of the switches can be fixed. Switches can be used solely for connecting the functional processing elements. Thus a parallel processor with a fixed interconnection structure can be fabricated. A wafer scale processor with a mesh, perfect shuffle, etc. interconnection topology can be implemented by embedding it within a wafer scale CHiP processor. The switch lattice simply remains in a static configuration.

Furthermore, the processing elements can be replaced by other components to implement a wafer scale system other than a parallel processor. For example, by replacing each PE by a 4K static RAM, a 3 Mbit wafer scale memory can be fabricated with existing technology [Egaw79, Lea79]. Additionally, the problems of address decoding, bit line driving, etc. must be solved, but the basic mechanism for connecting the individual storage modules can be based on the methodology for wafer scale CHiP processors.

3. Restructurable Design Methodology

Previously (section 2.5b) it was shown that redundancy can substantially reduce the manufacturing cost of a chip by increasing its yield. This suggests that building blocks with redundant components are useful on their own. A wafer can be scribed into the individual blocks which can be used as components of a larger system. The yield increase due to redundancy makes this a cost effective approach.

An alternate usage of redundancy is without changes in the fabrication technology to increase the maximum number of gates per chip. With fixed transistor size, wire width, etc., the integration level can be increased through the use of redundancy and restructurable circuitry. Furthermore, this design methodology (which was used for building blocks) can easily be generalized to apply to any system that can be divided into independent modules. These generalizations will be explored below.

There are three ways of increasing the number of components that can be fabricated on a single chip: increase chip area, improve circuit design, or reduce the size of the individual components. This work uses the first approach. The design methodology presented allows chips of larger area to be manufactured with acceptable yield.

What limits the size of a chip? Economics. It is prohibitively costly to manufacture very large chips. The manufacturing cost of a chip has three primary components.

total chip cost = processing cost + packaging cost + testing cost

As a first approximation, packaging and assembly costs are independent of the function performed by the chip, although they will increase slightly as the number of external connections to the chip increases. Similarly, test costs increase much more slowly than the complexity of the chip being tested, although sophisticated and high speed test equipment may be required. Thus, for larger and more complex chips, the packaging and test costs are approximately constant [Noyc77].

The cost of processing a wafer is independent of the number or type of chips patterned on it, so chip processing cost is proportional to the number of good chips to share the wafer cost. The cost of a chip then depends primarily on its yield. A typical yield curve (Figure 2.2.1) shows that yield declines quickly with increases in area. For large chips, the number of good chips drops rapidly pushing up their cost.

The exact yield at which point it is no longer feasible to manufacture a chip depends on the actual packaging, test and wafer processing costs. But for any fabrication process this point does exist, and it corresponds to a specific chip area. This is the *yield limit* of the technology. It is not economically feasible to fabricate chips of area larger than the yield limit. The fact that the yield declines quickly as a function of area causes a strict bound to be placed on the maximum allowable chip area. Exceeding this bound results in rapidly escalating chip cost. By reducing the rate of decline of Y, the yield limit will be extended allowing chips of larger area.

The cause of the rapid decline in yield is that a single defect renders the chip unusable. A defect may be introduced by any of the critical fabrication steps. It makes no difference in which step the defect is introduced, the end result is the same - a faulty chip. Consequently, in the yield equation (equation 2.2), there is a multiplicative effect of multiple processing steps; each step eliminates a fraction of the chips. The situation is analogous to tight rope walking - one slip and the game is over.

The slope of the yield curve can be lessened by decreasing s_0, the defect density, or the number of defect classes, k. In effect this introduces a more error free manufacturing process or reduces the number of fabrication steps. However, we have assumed a fixed technology. These modifications are not permitted. An alternative is to design fault tolerant chips. By introducing redundancy into the chip design, one or more defects can be absorbed, and the chip will still be functional.

What can be gained by designing chips with redundant modules? The maximum number of components per chip (which is determined by the maximum chip area since we have assumed fixed technology) is determined by the yield limit of the particular fabrication process. By adding redundant modules to this chip of maximum size, its yield can be increased resulting in lower cost (see Figure 7.3.1). Alternatively, by keeping cost constant, a more complex device can be fabricated. A device with yield below the yield limit can, through redundancy, have its yield increased to an acceptable level. In effect,

- use of redundancy allows the technology imposed yield limit to be surpassed.

The size and complexity of semiconductor devices spans a vast spectrum from SSI chips containing a few gates to wafer scale devices occupying vast amounts of silicon real estate (see Figure 7.3.2). Devices whose complexity and area surpass the yield limit are termed Ultra Large Area Chips or ULACs for short. They are not characterized by any absolute size since the position of the yield limit in the spectrum is technology dependent. The demarcation between conventional chips and ULACs is the requirement of fault tolerance to meet acceptable chip yield and cost.

(Note that 'ne concept of "acceptable" yield is inherently imprecise. Low yield (and hence high cost) may be acceptable for a new product commanding a premium price. Mature products facing competitive pressures may necessitate considerable higher yield.)

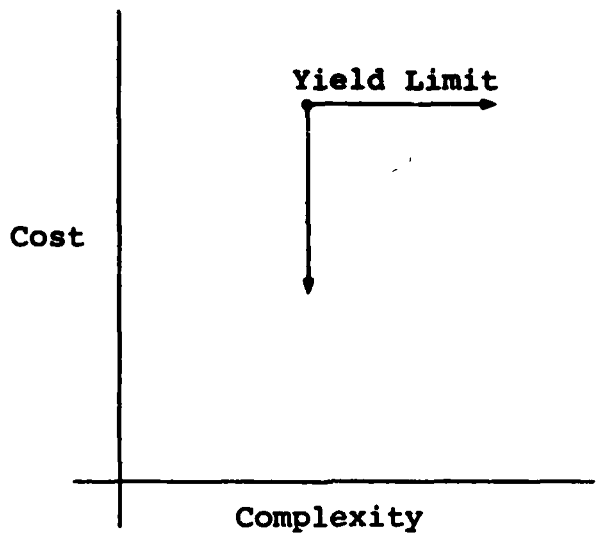


Figure 7.3.1 - Advantages of Restructurable Design Methodology

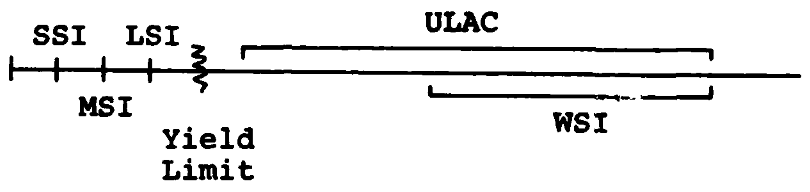


Figure 7.3.2 - Spectrum of Semiconductor Devices

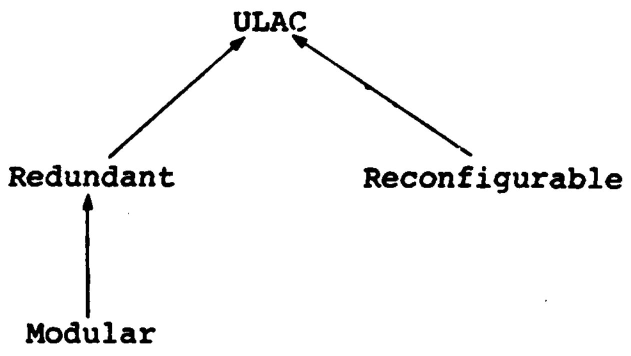


Figure 7.3.3 - Elements of the Restructurable Design Methodology

What are the design requirements in order to utilize redundancy? *Redundancy* necessitates a *modular* design. The system must be divided into separate and independent modules that can be replicated on the chip. Furthermore, only a small number of different module types are allowed. There must be spare copies of each different type of module. With many different types, the redundancy overhead becomes excessive, and the complexity of interconnecting the modules increases.

Since the occurrence of defects is a random process, it can not be known in advance which modules will be good and which will be bad. The pattern and number of faulty modules will vary from chip to chip. But it is necessary to connect together only the good modules. This requires a flexible means of interconnecting the modules. Furthermore, the interconnections between modules must be customized after the modules are completely fabricated and tested. In short, the circuitry must be *reconfigurable*. Mechanisms for implementing reconfiguration will be considered in the following section.

Modularity and reconfigurability are the key elements that enable redundancy to be utilized (see Figure 7.3.3). Through their combination, chips of larger area and hence greater complexity can be reliably and economically fabricated. These ultra large area chips offer substantial increases in integration level above the inherent limitations of fabrication technology.

Chips designed with the restructurable design methodology require overhead in the form of redundant modules and the wiring necessary to reconfigure the components. For this design methodology to be practical, this overhead must be limited. How can the overhead be kept to a reasonable level? First, it was noted (see Chapter 2) that higher module yield results in greater yield gains from redundancy. Thus modules with small area make more efficient use of silicon area and require lower overhead due to redundancy.

Second, since the reconfigurable wiring must at a bare minimum be capable of routing around a module, the wiring area is proportional to the square of the number of individual connections between modules. To reduce wiring overhead, it is necessary to limit the number of intermodule connections. Furthermore, to reduce the complexity of the wiring, a simple and regular pattern of connections between the good modules is required.

Note that the requirements of small modules with restricted and regular information flow are precisely those for designing algorithms for VLSI systems [Kung79]. The principles for integrated circuit design are the same as those required for the efficient implementation of restructurable, fault tolerant chips. There is a strong consonance between the restructurable design methodology and the general principles of good integrated circuit design. In fact, the restructurable design methodology may be considered to be a specialization and extension of the general design principles which has the added benefits of increasing the level of integration or, alternatively, reducing cost.

As a result, well designed chips can be relatively easily redesigned to employ reconfigurability and redundancy. Highly irregular circuitry will not naturally adapt to the requirement of modularity, and excessively complex designs may inherently require a large overhead for restructurable wiring. But simple, modular circuits can easily be extended for the addition of restructurable wiring between modules.

4. Future Research

This work gives rise to further questions concerning the performance and implementation of wafer scale CHiP processors. Some of the issues are: the design of a low impedance switch, the implementation of programmable power down capability, CMOS layout of PEs, etc. Perhaps of more general interest are the questions of larger scope concerning the extension of this work to restructurable circuitry and ultra large area chips. Two topics of particular interest are presented below.

a) Penalties for Restructurable Circuitry

The use of redundancy to increase the manufacturing yield of circuits is dependent on restructurable circuitry to provide flexible interconnections between modules. This yield increase is achieved at the expense of

- more modules per chip
- addition of extra interconnections
- an increase in signal delay
- computational effort in choosing the interconnection pattern for restructuring.

The first of these has been examined in some detail. The relationship between yield, redundancy and area was explored in Chapter 2. Secondly, additional wiring must be added to a chip to provide restructuring capability. Given that faults may occur in both the modules and the structurable wiring, how much wiring area must be provided to insure a high probability of restructuring? In addition to consuming chip area, restructurable wiring introduces longer wires between modules with resulting performance penalties. How much performance loss can be expected? What average wire lengths will exist between modules? In complex designs with many modules, choosing the best interconnection (or even finding an interconnection) may be a computationally difficult problem [Mann77, Aubu73]. Algorithms for restructuring homogeneous VLSI arrays also require further investigation.

b) Modular PE Design

The results of the analysis of redundancy (see Chapter 2) show that the highest leverage is obtained from the initial increments of redundancy. The first extra PE causes a large marginal increase in yield whereas successive redundant PEs cause smaller yield increases. Clearly, it is most area efficient to have a small degree of redundancy rather than a large amount.

In the wafer scale CHiP machine, switches and PEs are regarded as "black boxes" with no internal structure, and faulty building blocks are eliminated by brute force - column exclusion. All redundancy is within the building blocks, and the requirement for very high block yield forces a high degree of redundancy. Examining Figure 2.5.2 shows that N = 12 PEs is a very flat portion of the recovery curve. The addition of the 10th, 11th, and

12th PEs has increased recovery a total of only 1.7% (see Table 2.5.3).

A more efficient approach may be to have an extended hierarchy with additional levels and redundancy at more than one level. With a modest amount of redundancy introduced at several levels, very high yield for the topmost member of the hierarchy may be achieved with less area expenditure.

For example, one approach is to extend the hierarchy upwards. Building blocks are coalesced into super building blocks (SBBs). There are some redundant PEs and switches within each BB, and each SBB contains redundant building blocks. This combined redundancy can result in 99% yield of the SBBs which can then be composed using column exclusion.

The problem with this approach is that higher up in the hierarchy the number of connections between units increases. For example, in the wafer scale CHiP processor, there are ten connections between a pair of switches, but connecting two building blocks requires 90 wires. Since blocks within each SBB must be flexibly interconnected, a switching structure to connect blocks must be provided. With switch area proportional to the square of the number of connections, a single switch routing 90 wires occupies a large area and consequently has low yield. Instead of a single large switch, routing can be implemented with a large number of small switches. However, this substantially increases the number of switching levels between PEs resulting in reduced preformance. In short, there is no practical method of extending the hierarchy upward.

An alternate solution is to extend the hierarchy downward. Instead of treating PEs as individual units, impose a modular and reconfigurable structure on the individual PEs. By dividing them into independent modules, placing redundant modules within each PE and reconfigurable wiring between modules, PE yield can be substantially increased. Increasing PE yield reduces the redundancy required within each block. Increasing PE yield from the current 65% for the "standard" PE to 80% reduces the number of PEs per block from 12 to 8 while still maintaining 99% block yield.

Memory redundancy is easily incorporated into each PE using standard techniques with spare rows (or columns) in the memory array [Smit81, Kokk81, Mano80]. There are two ways of dividing the datapath of the PE into modules: slice "horizontally" dividing into bit slices or slicing "vertically" creating pipelined segments.

The bit slice modularization is easy to design; each module is a miniature version of the original datapath. Pipelining provides the potential for increased performance by each PE but is more difficult to design. Since one module may be substituted for a faulty module, all modules must have identical hardware. But each stage in an arithmetic pipeline performs a different operation so the modules must be microcoded to specialize them for a particular position in the pipeline.

A topic for future research is to design PE modules which are flexible, powerful and of acceptable size. For a particular processing element, comparison of the bit slice and pipelined approaches will shed light on the area - performance - yield tradeoffs of different modularizations.

The restructurable wiring within each PE will introduce delays into the basic cycle time of the PE. A programmable switching structure may introduce an unacceptable performance penalty. An alternative is to use permanent links to reconfigure the modules [Smit81, Kokk81, Logu80]. The loss of flexibility is balanced by a decrease in connection impedance. The feasibility of the modular approach depends in part on the performance loss due to restructuring and the extent to which it can be balanced by pipelining.

BIBLIOGRAPHY

Bibliography

Aubu73	Aubusson,R. AND Catt,I. "Wafer-Scale Integration - A Fault - Tolerant Procedure," *IEEE J. Solid-State Circuits*. **SC-13** ,3 (June 1973), 339-344.
Aubu78	Aubusson,R.C. AND Gledhill,R.J. "Wafer-Scale Integration - Some Approaches to the Interconnection Problem," *Microelectronics* **V-9** , 1 (Jan. 1978), 5-10.
Batc79	Batcher,K. "MPP - A Massively Parallel Processor," *Proc. of 1979 International Conf. on Parallel Processing*, (Aug. 1979), 249.
Blak75	Blakeslee,T.R. *Digital Design with Standard MSI and LSI*, Wiley, New York, 1975.
Budz82	Budzinski,R., Linn,J. AND Thatte,S. "A Restructurable Integrated Circuit for Implementing Programmable Digital Systems," *IEEE Computer*. **V-15** ,3 (March 1982),43-54.
Calh72	Calhoun,D.F. AND McNamee,L.P. "A Means of Reducing Custom LSI Interconnection Requirements," *IEEE J. Solid-State Circuits*. **SC-7** ,5 (Oct. 1972), 395-404.
Cenk79	Cenker,R.P., *et. al.*, "A Fault-Tolerant 64K Dynamic Random-Access Memory," *IEEE Trans. Electron. Devices* **ED-26** ,6 (June 1979),853-860.
Chap	Chapman,G. private communication, Lincoln Lab.
Cuny82	Cuny,J. AND Snyder,L. "Testing Coordination for "Homogeneous" Parallel Algorithms," *Proc. of 1982 International Conf. on Parallel Processing*, (Aug. 1982).
DasG78	DasGupta,S., Eichelberger,E. AND Williams,T.W. "LSI Chip Design for Testability," *ISSC 1978* , 216-217.
DeSi79	DeSimone,R.R., Donofrio,N.M., Flur,B.L., Kruggel,R.H., Leung,H.H., AND Schnadt,R. "FET RAMs," *1979 IEEE ISSCC Digest of Tech. Papers* **22** ,(1979), 154-155.
Eato81	Eaton,S.S. "A 100ns 64K Dynamic RAM Using Redundancy," *ISSCC Dig. of Tech. Papers* (1981), 84-85.

Egaw79 Egawa,Y., Tsuda,N. AND Masuda,K. "A 1Mb Full Wafer MOS RAM," *ISSCC Dig. of Tech. Papers* (1979), 18-19.

Elme77 Elmer,B.R. *et. al.* "Fault Tolerant 92160 Bit Multiphase CCD Memory," *ISSCC Dig. of Tech. Papers* (1977), 116-117.

Fitz80 Fitzgerald,B.F. AND Thoma,E.P. "Circuit Implementation of Fusible Addresses on RAMS for Productivity Enhancement," *IBM J. of Res. and Dev.* **V-24**, 3 (May 1980), 291-298.

Fitz81 Fitzpatrick,D.T., *et. al.* "VLSI Implementation of a Reduced Instruction Set Computer," *CMU Conf. on VLSI* (1981), 327-336.

Fuss82 Fussell,D. AND Varman,P. "Fault - Tolerant Wafer - Scale Architectures for VLSI," *Proc. 9th Annual Symp. on Computer Architecture* (1982), 190-198.

Gann81 Gannon,D. AND Snyder,L. "Linear Recurrence Systems for VLSI: The Configurable, Highly Parallel Approach," *Proc. of the 1981 International Conf. on Parallel Processing*, (Aug. 1981), 259-260.

Gare79 Garey, M.R. AND Johnson, D.S. *Computers and Intractability A Guide to the Theory of NP - Completeness*, W.H. Freeman, San Francisco, 1979.

Glas79 Glaser,A.B. AND Subak-Sharpe,G.E. *Integrated Circuit Engineering* , Addison-Wesley, Reading, 1979.

Gupt72 Gupta,A. AND Lathrop,J.W. "Yield Analysis of Large Integrated-Circuit Chips," *IEEE J. Solid-State Circuits.* **SC-7** ,5 (Oct. 1972), 389-395.

Haye80 Hayes,J.F. AND McCluskey,E.J. "Testability Considerations in Microprocessor-Based Design," *IEEE Computer* **V-13** ,3 (March 1980), 17-26.

Hedl81a Hedlund,K.S. "Design of a Prototype Blue CHiP Processing Element," Tech. Report 388, Comp. Sci. Dept., Purdue Univ., June 1981.

Hedl81b Hedlund,K.S. AND Snyder,L. "A Model for Wafer Scale Testing," Tech. Report 389, Comp. Sci. Dept., Purdue Univ., Sept. 1981.

Hedl82a Hedlund,K.S. AND Snyder,L. "Wafer Scale Integration of Configurable, Highly Parallel CHiP Processors," Tech. Report 407 Comp. Sci. Dept., Purdue Univ., April 1982.

Henn81 Hennessy,J., Jouppi,N., Baskett,F. AND Gill,J. "MIPS: A VLSI Processor Architecture," *CMU Conf. on VLSI* (1981), 337-346.

Hon80 Hon,R.W. AND Sequin,C.H. *A Guide to LSI Implementation* 2nd Edition, Xerox (Jan. 1980).

Hsia82 Hsiao,C. "Highly Parallel Processing of Relational Databases," Ph.D. Dissertation, Comp. Sci. Dept., Purdue U., Aug. 1982.

IEEE82 International Eletrical and Electonics Engineers "Whatever Happened to Wafer-Scale Integration?" *IEEE Spectrum*, **V-19** , 6(June 1982), 18.

Klei81 Kleitman,D. *et. al.* "New Layouts for the Shuffle-Exchange Graph (Extended Abstract)," *Symp. on Theory of Computing* (May 1981), 278-292.

Kokk81 Kokkonen,K., *et. al.* "Redundancy Techniques for Fast Static RAMs," *ISSCC Dig. of Tech. Papers* (1981), 80-1.

Koni82 Konishi,S. "A 64Kb CMOS RAM," *ISSCC Dig. of Tech. Papers* (1982), 258-259.

Kore81 Koren, I. "A Reconfigurable and Fault - Tolerant VLSI Multiprocessor Array," *Proc. 8th Annual Symp. on Computer Architecture* (1981), 425-442.

Kuhn75 Kuhn,L. "Experimental Study of Laser Formed Connections for LSI Wafer Personalization," *IEEE J. of Solid-State Circuits* **SC-10** ,4 (Aug. 1975), 219-228.

Knut70 Knuth,D.E. "An Empirical Study of FORTRAN Programs," *Software - Practice and Experience*, **V-1** ,11 (Nov. 1970), 105-133.

Kung79 Kung,H.T. "Let's Design Algorithms for VLSI Systesm," *Proc. of Caltech Conf. on Very Large Scale Integration*, (Jan. 1979),65-90.

Kung82 Kung,H.T. "Why Systolic Arcitectures?" *IEEE Computer*, **V-15** ,1 (Jan. 1982), 37-46.

LaPa78a LaPaugh, A.S. "The Subgraph Homeomorphism Problem," Tech. Memo 99, Lab. for Comp. Sci., MIT, Feb. 1978.

LaPa78b LaPaugh, A.S. AND Rivest, R.L. "The Subgraph Homeomorphism Problem," *Proc. 10th Annual Symp. on Theory of Computing* (1978), 40-50.

Laws66 Lawson,T.R. "A Prediction of the Photoresist Influence on Integrated Circuit Yield," *Semicond. Prod. Solid State Technol.* **V-9** ,7 (July 1966), 22-25.

Lea79 Lea,R.M. AND Streetharan,M. "WSI Distributed Logic Memories," *Proc. Caltech Conf on Very Large Scale Integration* (Jan. 1979), 187-197.

Logu80 Logue,J.C. *et. al.* "Techniques for Improving Engineering Productivity of VLSI Designs," *Proc. of IEEE International Conf. on Circuits and Computers* (1980), 248-251.

Mann77 Manning,F. "An Approach to Highly Integrated, Computer-Maintained Cellular Arrays," *IEEE Trans. Comput.* **C-26** , 6 (June 1977), 536-552.

Mano80 Mano,T. "A 256K RAM Fabricated With Molybdenum - Ploysilicon Technology," *ISSCC Dig. of Tech. Papers* (1980), 234-235.

Mead80 Mead,C. AND Conway,L. *Introduction to VLSI Systems* , Addison-Wesley, Reading, 1980.

Mina81 Minato,O., Masuhara,T., Sasaki,T., Sakai,Y. AND Yoshzaki,K. "A High-Speed Hi-CMOSII 4K Static RAM," *IEEE J. Solid-State Circuits* **SC-16** ,5 (Oct 1981), 449-454.

Mina82 Minato, *et.al.* "A HI-CMOSII 8K × 8b Static RAM," *ISSCC Dig. of Tech. Papers*, (1982), 256-257.

Mina80 Minato,O., Masuhara,T. AND Sakai,Y. "HI-CMOS 4K Static RAM," *ISSCC Dig. of Tech. Papers* (1980),234-5.

Moor79 Moore,G.E. "Are We Really Ready for VLSI?" *Proc. of Caltech Conf. on Very Large Scale Integration* (Jan. 1979), 3-14.

Nair78 Nair,R., Thatte,S.M. AND Abraham,J.A. "Efficient Algorithms for Testing Semiconductor Random - Access Memories," *IEEE Trans. Comput.* **C-27** ,6 (June 1978), 572-576.

Noyc77 Noyce,R.N. "Large Scale Integration: What is Yet to Come?" *Science* , (March 18, 1977), 1102-1106.

Owen81 Owens, M.R. "Compound Algorithms for Digit Online Algorithms," Tech. Reprot CS-81-1, Comp. Sci. Dept., Penn. State U., Jan. 1981.

Parz60 Parzen,E. *Modern Probability Theory and Its Applications*, Wiley, New York, 1960.

Patt81 Patterson,D.A. AND Sequin,C.H. "RISC I: A Reduced Instruction Set VLSI Computer," *Proc. 8th Annual Symp. on Computer Architecture* (1981), 443-457.

Petr67 Petritz,R.L. "Current Status of Large Scale Integration Technology," *IEEE J. Solid-State Circuits.* **SC-2** ,4 (Dec. 1967), 130-147.

Peut77a Peuto,B.L. AND Shustek,L.J. "Current Issues in the Architecture of Microprocessors," *IEEE Computer*, **V-10** ,2 (Feb. 1977),20-25.

Peut77b Peuto,B.L. AND Shustek,L.J. "An Instruction Timing Model of CPU Performance," *Proc. 4th Annual Symp. on Computer Architecture*, (1977), 165-178.

Phis79 Phister,M. "Technology and Economics: Integrated Circuit Manufacturing Costs," *Computer Design*, **V-18**, 10 (Oct. 1979), 34-42.

Pric70 Price, J. E. "A New Look at Yield of Integrated Circuits," *Proc. IEEE.* **20** (May 1976), 228-234.

Raff79 Raffel,J.I. "On the Use of Nonvolatile Programmable Links for Restructurable VLSI," *Proc. of Caltech Conf. on Very Large Scale Integration*, (Jan. 1979),95-104.

Rees81 Reese,E.A. *et. al.* "A 4K × 8 Dynamic RAM With Self Refresh," *ISSCC Dig. of Tech. Papers* (1981), 88-89.

Ross76 Ross,S. *A First Course in Probability* , Macmillan, New York, 1976.

Radi82 Radin,G. "The 801 Minicomputer," *Symp. on Arch. Support for Prog. Langs. and Operating Sys.* (March 1982), 39-47.

Rung81 Rung,R.D. "Determining IC Layout Rules for Cost Minimization," *IEEE J. Solid-State Circuits* SC-16 , 1 (Feb. 1981), 35-43.

Sait82	Saito,K. AND Arai,E. "Experimental Analysis and New Modeling of MOS LSI Yield Associated with the Number of Elements," *IEEE J. Solid-State Circuits* **SC-17** , 1 (Feb. 1982), 28-33.
Seit79	Seitz, C.L. "Self - Timed VLSI Systems," *Proc. Conf. on Very Large Scale Integration: Architecture, Design and Fabrication* (1979).
Smit81	Smith,R.T. *et. al.* "Laser Programmable Redundancy and Yield Improvements in a 64K DRAM," *IEEE J. of Solid-State Circuits* **SC-16** , 5 (Oct. 1981), 506-514.
Snyd82a	Snyder,L. "Introduction to the Configurable, Highly Parallel Computer," *IEEE Computer* **V-15** , 1 (Jan. 1982), 47-56.
Snyd82b	Snyder,L. "Configurable, Highly Parallel (CHiP) Approach for Signal Processing Applications," *Proc. Tech. Symp. East '82*, SPIE, 1982.
Stap73	Stapper,C.H. "Defect Density Distribution for LSI Yield Calculations," *IEEE Tran. Electron Devices* **ED-20** ,7 (July 1973), 655-657.
Stap75	Stapper,C.H. "On a Composite Mocel to the IC Yield Problem," *IEEE J. Solid-State Circuits* **SC-10** , 6 (Dec. 1975),537-539.
Stap76	Stapper,C.H. "LSI Yield Modeling and Process Monitoring," *IBM J. Res. Dev.* **V-20** , 3 (May 1976), 228-234.
Stap80	Stapper, C.H., McLaren,A.N., AND Dreckmann,M. "Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product," *IBM J. Res. and Dev.* **V-24** ,3 (May 1980),398-409.
Stap82	Stapper,C.H. "Yield Models for 256K RAMs and Beyond," *ISSCC Dig. of Tech. Papers* (1982), 12-13.
Stee81	Steele,T.S. "Terminal and Cooling Requirements for LSI Packages," *IEEE Trans. on Components, Hybrids and Manufacturing Tech.* , (June 1981),187-191.
Suth77	Sutherland,I.E. AND Mead,C.A. "Microelectronics and Computer Science," *Scientific American* **V-237** , 3 (Sept. 1977), 210-228.
West	Weste,N. private communication, U. of North Carolina.

Will73 Williams,M.J.Y. AND Angell,J.B., "Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic," *IEEE Trans. Comput.* **C-22** ,1 (Jan. 1973), 46-60.

Will79 Williams,T.W. AND Parker,K.P. "Testing Logic Networks and Designing for Testability," *IEEE Computer* **V-12** ,10 (Oct. 1979), 9-21.

Wu82 Wu,W. "Automated Welding Customizes Programmable Logic Arrays," *Electronics* **V-55** , 14 (July 14,1982), 159-162.

Yu81 Yu,K., Chwang,J.C., Bohr,M., Warkentin,P., Stern,S. AND Berglund,C.N., "HMOS-CMOS - A Low-Power High-Performance Technology," *IEEE J. Solid-State Circuits* **SC-16** , 5 (Oct. 1981), 454-459.

APPENDIX

APPENDIX 1

SUMMATION OF RANDOM VARIABLES

In this appendix we derive the probability

P' = P'(i, nf, Np) = Pr(i defects occupy nf or fewer of Np PEs)

where Np is the total number of PEs in a sublattice which contains nf redundant PEs and where i > nf. The i defects all fall in a set of Np PEs. P' is the probability that the defects occupy a subset of size nf or smaller. The form of P' varies depending on the assumptions which are made about the processing technology. As the assumptions are made more realistic, the analytical form of P' can become very cumbersome. P' will first be derived under a simple set of assumptions, and the results will be progressively refined.

The Price model assumes distinguishable classes of indistinguishab defects. For the first approximation, assume only one class so that all defects are indistinguishable. This corresponds to lumping the effect of all processing steps and regarding the wafer to be manufactured in a single step. We do not differentiate between defects introduced at different stages of the fabrication process. This model is called the *lumped approximation*.

It is simple to derive and is a useful first approximation.

1. Lumped Approximation

It is tempting to try to evaluate P' by

$$P' = \sum_{k=1}^{nf} \binom{Np}{k} \text{ Pr(i defects occupy k PEs)} \qquad (1.1)$$

However, this is somewhat ambiguous and leads to difficulties. For instance, consider the number of different possible assignments of 4 defects to 3 PEs. This includes some assignments in which 2 of the PEs each contain 2 defects and the third PE is defect free. Only 2 of the 3 PEs contain any defects at all. This assignment is already counted when placing 4 defects in 2 PEs. Therefore, equation 1.1 double counts many assignments. To avoid double counting, we will be more precise in our terminology. We will say i defects *fall in* k PEs if the defects occupy k or fewer PEs; some of the k PE may be defect-free. i defects ***cover*** k PEs if the defects fall in k PEs, and every PE contains at least one defect.

We can correctly restate equation 1.1

$$P' = \sum_{k=1}^{nf} \binom{Np}{k} \text{ Pr(i defects cover k PEs)} =$$

$$\sum_{k=1}^{nf} \binom{Np}{k} \begin{matrix}\text{(number of placements of i defects which cover k} \\ \text{PEs)/(total number of placements of i defects} \\ \text{in Np PEs)}\end{matrix} \qquad (1.2)$$

Since there are $\binom{i+w-1}{i}$ different ways of placing i indistinguishable defects in w PEs [Ross76], there are

$$\binom{i+Np-1}{i}$$

different placements of i defects in Np PEs.

For any particular subset of k PEs, how many of these placements cover the subset? First, take k of the i defects and assign one to each PE of the subset. This insures that the subset is covered. The remaining i−k defects can be assigned to any of the k PEs. There are

$$\binom{(i-k)+k-1}{i-k} = \binom{i-1}{i-k}$$

ways of doing this. This completes the lumped approximation

$$P' = \sum_{i=1}^{nf} \binom{Np}{k} \frac{\binom{i-1}{i-k}}{\binom{i+Np-1}{i}} = \frac{1}{\binom{i+Np-1}{i}} \sum_{k=1}^{nf} \binom{Np}{k} \binom{i-1}{i-k} \qquad (1.3)$$

A more accurate approximation can be derived by modeling more than one fabrication step [Glas79]. This introduces multiple, distinguishable classes of indistinguishable defects. Each individual class follows a lumped approximation, but the fact that i defects can be partitioned into multiple classes in many different ways must be accounted for.

The first results derived will be for 2 classes of defects. A more realistic model for Blue CHiP applications is a four class model. The 3 and 4 class formulae will be derived in a manner similar to the 2 class derivation.

Figure A1.1 shows P'(4,i,16), the probability that is defects all fall in 4 or fewer of 16 PEs, for the lumped, 2-class and 3-class solutions.

2. Two Class Approximation.

In refining the lumped approximation, the following assumptions will be made:

1) There are two distinguishable classes of indistinguishable defects. Each class represents a separate fabrication step.

2) The fabrication steps are independent.

3) The total number of defects is the sum of the defects introduced at each step.

4) There is an equal probability of a defect belonging to either class.

Given that there are exactly i_1, defects of class 1 and i_2 of class 2, consider the probability that the total number of defects, $i = i_1 + i_2$, fall in nf or fewer of Np PEs. This quantity is denoted by Q''. To evaluate Q'', we condition on k, the number of defects covered by defects of both classes.

$$Q'' = \sum_{k=1}^{nf} \binom{Np}{k} \text{ Pr(i defects cover a set of k PEs)} \qquad (2.1)$$

For any particular set of k PEs,

$$\text{Pr(i defects cover set)} = \text{(numb placements of } i_1 \text{ and } i_2 \text{ that cover set)/} \\ \text{(total numb placements of } i_1 \text{ and } i_2 \text{ in Np PEs)} \qquad (2.2)$$

Consider the denominator of the above equation. Since the fabrication steps are independent,

$$\text{total number of placements of } i_1 \text{ and } i_2 \text{ in Np PEs} = \\ \text{(number of placements of } i_1 \text{ in Np PEs)} * \\ \text{(number of placements of } i_2 \text{ in Np PEs)} =$$

$$\binom{i_1 + Np - 1}{i_1} \binom{i_2 + Np - 1}{i_2}$$

This quantity will be denoted by Place (i_1, i_2; Np) with the obvious extension to Place (i_1, ..., i_N; Np) following from the independence of all processing steps.

To evaluate the numerator of equation 2.2 we condition on the number of different PEs in the set of size k occupied by class 1 defects.

$$\text{numerator} = \sum_{c_1} \binom{k}{c_1} \text{(numb placements of } i_1 \text{ that cover } c_1 \text{ PEs)}$$
$$\text{(numb placements of } i_2 \text{ that occupy } k-c_1 \text{ remaining PEs)}$$

For any subset of size c_1, select c_1 of the class 1 defects and place one defect in each PE of the subset. This insures that all members of the subset are occupied. The remaining $i_1 - c_1$ defects can be distributed over the c_1 PEs in

$$\binom{(i_1-c_1) + c_1-1}{i_1 - c_1} = \binom{i_1-1}{i_1-c_1}$$

different ways. There are $k-c_1$ members of the set not covered by defects of the first class. Therefore, these PEs must be occupied by class 2 defects. We take $k-c_1$ of the i_2 class 2 defects and put one in each of the PEs not covered by class 1. This insures that the entire set of k PEs is covered. The remaining $i_2-(k-c_1)$ defects can be distributed amongst any of the k PEs. Consequently, there are

$$\binom{(i_2 - (k-c_1)) + k-1}{i_2 - (k-c_1)} = \binom{i_2 + c_1-1}{i_2 + c_1-k}$$

ways of placing the class 2 defects to insure that all the k PEs contain at least one defect. Consequently, there are

$$\binom{i_1-1}{i_1-c_1}\binom{i_2+c_1-1}{i_2+c_1-k}$$

different ways of placing the i_1 and i_2 to cover the subset. We will denote this quantity by Cover (i_1, i_2, c_1, k).

This completes the evaluation of the numerator of equation 2.2,

$$\text{numerator} = \sum_{c_1} \binom{k}{c_1} \text{Cover}\ (i_1, i_2; c_1, k)$$

To evaluate the limits of the summation,[1] note that the class 1 defects can cover at most i_1 of the k PEs. Furthermore, the class 1 defects must cover at least 1 PE (unless there are no class 2 defects). The class 2 defects must occupy the remaining $k-c_1$ PEs not covered by the class 1 defects. So $i_2 \geq k-c_1$ or $c_1 \geq k-i_2$. By introducing a one argument form of the Kronecker delta function

$$\delta_0(i) = \delta(i,0) = \begin{cases} 0 & i=0 \\ 1 & i>0 \end{cases}$$

we have

$$\text{numerator} = \sum_{c_1=\max(\delta_0(i_2),k-i_2)}^{\min(i_1,k)} \binom{k}{c_1} \text{Cover}\ (i_1, i_2; c_1, k)$$

[1] We assume $\binom{a}{b} = 1$ for $a<b$ or $a<0$ or $b<0$.

This completes the evaluation of equation 2.2 and

$$Q'' = \sum_{k=1}^{nf} \binom{Np}{k} \frac{\sum_{c_1} \binom{k}{c_1} \mathrm{Cover}(i_1, i_2, ; c_1, k)}{\mathrm{Place}(i_1, i_2, i_3; Np)}$$

with the limits for c_1 as above.

Now, Q" assumes there are exactly i_1 and i_2 defects of each class. We can use P" to evaluate

$$Q'' = \Pr(i \text{ defects fall in nf or fewer of Np PEs}) =$$

$$\sum_{i_1+i_2=i} Q'' \Pr(i \text{ defects are partitioned with } i_1 \text{ AND } i_2 \text{ in each class}) =$$

$$= \sum_{i_1+i_2=i} Q'' \, \mathrm{Part}(i; i_1, i_2) \tag{2.3}$$

To evaluate the partition function, Part, let I_1 and I_2 be random variables representing the number of defects in each class and i be the total number of defects. Consider the partitioning of defects into two classes to be an experiment i trials with each trial deciding which class a defect will be in. The partitioning of a fixed number of defects into two classes then follows a binomial distribution [Ross76].

$$\Pr(I_1 = i_1) = \binom{i}{i_1} p^{i_1}(1-p)^{i-i_1}$$

Since it is equally likely that a defect will be in either class (by assumption four above), we have $p=\frac{1}{2}$ and

$$\Pr(I_1=i_1) = \begin{pmatrix} i \\ i_1 \end{pmatrix} (1/2)^{i_1} (1 = 1/2)^{i-i_1} = \frac{1}{2^i} \begin{pmatrix} i \\ i_1 \end{pmatrix}$$

Since I_1 and I_2 must sum to i,

$$\Pr(I_1 = i_1 \text{ AND } I_2 = i-i_1) = \Pr(I_1 = i_1) = \text{part}(i;\, i_1,\, i-i_1) = \frac{1}{2^i} \begin{pmatrix} i \\ i_1 \end{pmatrix}$$

This completes the evaluation of the two class approximation with equation 2.3 becoming

$$P'' = \sum_{i_1+i_2=i} \text{Part}\,(i;\, i_1,\, i_2) \sum_{k=1}^{nf}$$

$$\begin{pmatrix} Np \\ k \end{pmatrix} \frac{\displaystyle\sum_{c_1=\max(\delta_0)i_2),\, k-i_2)}^{\min(i_1,k)}{}' \begin{pmatrix} k \\ c_1 \end{pmatrix} \text{Cover}\,(i_1,\, i_2;\, c_1,\, k)}{\text{Place}\,(i_1,\, i_2;\, Np)} \tag{2.4}$$

3. Extension to Three Classes

The derivation under the assumption of three distinguishable classes of defects is similar to the 2-class case. P''' will denote the probability under the 3-class assumption. By a simple extension of the 2-class derivation.

$$P''' = \sum_{i_1+i_2+i_3=i} \text{Part}\,(i;\, i_1,\, i_2,\, i_3) \sum_{k=i}^{nf}$$

$$\begin{pmatrix} k \\ Np \end{pmatrix} \Pr(i_1,\, i_2 \text{ and } i_3 \text{ cover the set}) \tag{3.1}$$

and we can decompose this last probability for a specific set of the PEs.

$\Pr(i_1,\, i_2$ and i_3 cover the set) = (number of placements of i_1, i_2 and i_3

that cover the set)/ Place $(i_1, i_2, i_3; Np)$ (3.2)

where the three argument versions of Part and Place are simple extensions of the two argument functions:

A) Place. By the independence of the processing steps

$$\text{Place}\,(i_1, i_2, i_3; Np) = \binom{i_1+Np-1}{i_1}\binom{i_2+Np-1}{i_2}\binom{i_3+Np-1}{i_3} =$$

$$\binom{i_1+Np-1}{i_1}\,\text{Place}\,(i_2, i_3; Np)$$

B) Part. We define

Part $(i; i_1, i_2, i_3)$ = probability that i defects are partitioned with i_1, in class 1, i_2 in class 2 AND i_3 in class 3

$= \Pr(I_1 = i_1 \text{ AND } I_2 = i_2 \text{ AND } I_3 = i_3)$

$= \Pr(I_1 = i_1)\Pr(I_2 = i_2 \mid I_1 = i_1)$

where I_1, I_2 and I_3 are random variables representing the number of defects in each class. Note that the number of defects in class 3 need not be explicitly accounted for. Since $i = i_1 + i_2 + i_3$, choosing i_1 and i_2 determines i_3.

It is equally likely that a defect will be in anyone of the three classes. Therefore, $\Pr(I_1 = i)$ follows a binomial distribution

$$\Pr(I_1 = i_1) = \binom{i}{i_1}\left(\frac{1}{3}\right)^{i_1}\left(1-\frac{1}{3}\right)^{i-i_1} = \frac{2^{i-i_1}}{3^i}\binom{i}{i_1} \tag{3.3}$$

Now, the conditional portion of $\Pr(I_2 = i_2 \mid I_1 = i_1)$ constrains the remaining $i-i_1$ defects to fall in either class 2 or class 3. Both are equally probably, so once again a binomial distribution is followed

$$Pr(I_2 = i_2 \mid I_1 = i_1) = \binom{i-i_1}{i_2}(1/2)^{i_2}(i-1/2)^{i-i_1-i_2} =$$

$$\frac{1}{2^{i-i_1}}\binom{i-i_1}{i_2} \tag{3.4}$$

Combining equations 3.3 and 3.4 gives

$$\text{Part}\ (i;\ i_1,\ i_2,\ i_3) = \left[\frac{2^{i-i_1}}{3^i}\right]\binom{i}{i_1}\frac{1}{2^{i-i_1}}\binom{i-i_1}{i_2} =$$

$$\frac{1}{3^i}\binom{i}{i_1}\binom{i-i_1}{i_2} = \frac{1}{3^i}\,\frac{i!}{i_1!\,i_2!\,i_3!}$$

The evaluation of P''' is now complete except for the numerator of equation 3.2 which is evaluated as in the 2-class situation, but with an additional summation required due to the additional class.

numerator = number of placements of i_1, i_2, i_3 which cover a set of k PEs =

$$\sum_{c_1}\binom{k}{c_1}\binom{(i_1-c_1)+c_1-1}{i_1-c_1}\begin{array}{l}\text{(number of placements of } i_2 \text{ and } i_3 \text{ which}\\ \text{occupy } k-c_1 \text{ remaining PEs)}\end{array} \tag{3.5}$$

Given a particular subset of size c_1, we calculate as follows the number of placements of i_2 and i_3 that insure the set of k PEs is covered. Condition on c_2, the number of previously defect free PEs occupied by class 2 defects.

number of placements of i_2 and i_3 which occupy $k-c_1$ remaining PEs =

$$\sum_{c_2}{}'\binom{k-c_1}{c_2}\begin{array}{l}\text{(number of placements of } i_2 \text{ whichoccupy } c_2 \text{ previously defect}\\ \text{free PEs) (number of placements of } i_3 \text{ which occupy}\\ k-c_1-c_2 \text{ remaining PEs)}\end{array} \tag{3.6}$$

Select c_2 of the i_2 class 2 defects and place each in a PE not already occupied by a class 1 defect. This insures that exactly c_1 and c_2 different PEs are covered by classes 1 and 2. The remaining i_2-c_2 class 2 defects can be assigned to any of the c_1 and c_2 PEs already covered. There are

$$\binom{(i_2-c_2) + (c_1+c_2)-1}{i_2-c_2} = \binom{i_2+c_1-1}{i_2-c_2} \tag{3.7}$$

different ways of making the class 2 assignments.

Similarly, $k-c_1-c_2$ class 3 defects are required to complete the covering of the set of k PEs. The remaining $i_3-(k-c_1-c_2)$ class 3 defects can be assigned to any of the k PEs in

$$\binom{i_3-k+c_1+c_2) + k-1}{i_3-k+c_1+c_2} = \binom{i_3+c_1+c_2-1}{i_3+c_1+c_2-k} \tag{3.8}$$

different ways.

Substituting equations 3.7 and 3.8 back into equation 3.6, number of placements of i_2 and i_3 which cover $k-c_1$ PEs is

$$\sum_{c_2} \binom{k-c_1}{c_2} \binom{i_2+c_1-1}{i_2-c_2} \binom{i_3+c_1+c_2-1}{i_3+c_1+c_2-k} \tag{3.9}$$

To determine the limits of the summation, note that the class 2 defects must occupy at least 1 PE (unless there are no class 2 defects). Furthermore, the class 3 defects must cover the remaining $k-c_1-c_2$ PEs not covered by classes 1 and 2 so $i_3 \geq k-c_1-c_2$ or $c_2 \geq k-c_1-i_3$. Consequently, c_2 assumes values from max $(\delta_0(i_2), k-c_1-i_3)$ to min $(i_2, k-c_1)$.

To simplify notation, we introduce a three argument version of the Cover function

$$\text{Cover}\ (i_1, i_2, i_3; c_2, k) = \binom{i_1-i}{i_1-k} \sum_{\substack{C_2=\max(\delta_0(i_2),\\ k-c_1-i_3)}}^{\min(i_2, k-c_1)}$$

$$\binom{k-c_1}{c_2}\binom{i_2+c_1-1}{i_2-c_2}\binom{i_3+c_1+c_2-1}{i_3+c_1+c_2-k}$$

So, for a specific set of k PEs, equation 3.2 can be rewritten

$$\Pr(i_1,\ i_2 \text{ and } i_3 \text{ cover the set}) =$$

$$\frac{1}{\text{Place }(i_1-i_3;\ Np)} \sum_{\substack{c_1=\max(\delta_0(i1),\\ k-i_2-i_3)}}^{\min(i_1,k)} \binom{k}{c_1} \text{Cover }(i_1-i_3;\ c_1,\ k)$$

where the limits for c_1 are derived similarly to c_2. Finally, we can write P''' as

$$P''' = \Pr(\text{i defects occupy nf or fewer of Np PEs}) =$$

$$\sum_{i_1+i_2+i_3=i} \text{Part }(i;\ i_1-i_3) \sum_{k=1}^{nf} \binom{Np}{k} \frac{\sum_{c_1} \binom{k}{c_1} \text{Cover}(i_1-i_3;\ c_1,\ k)}{\text{Place }(i_1-i_3;\ Np)}$$

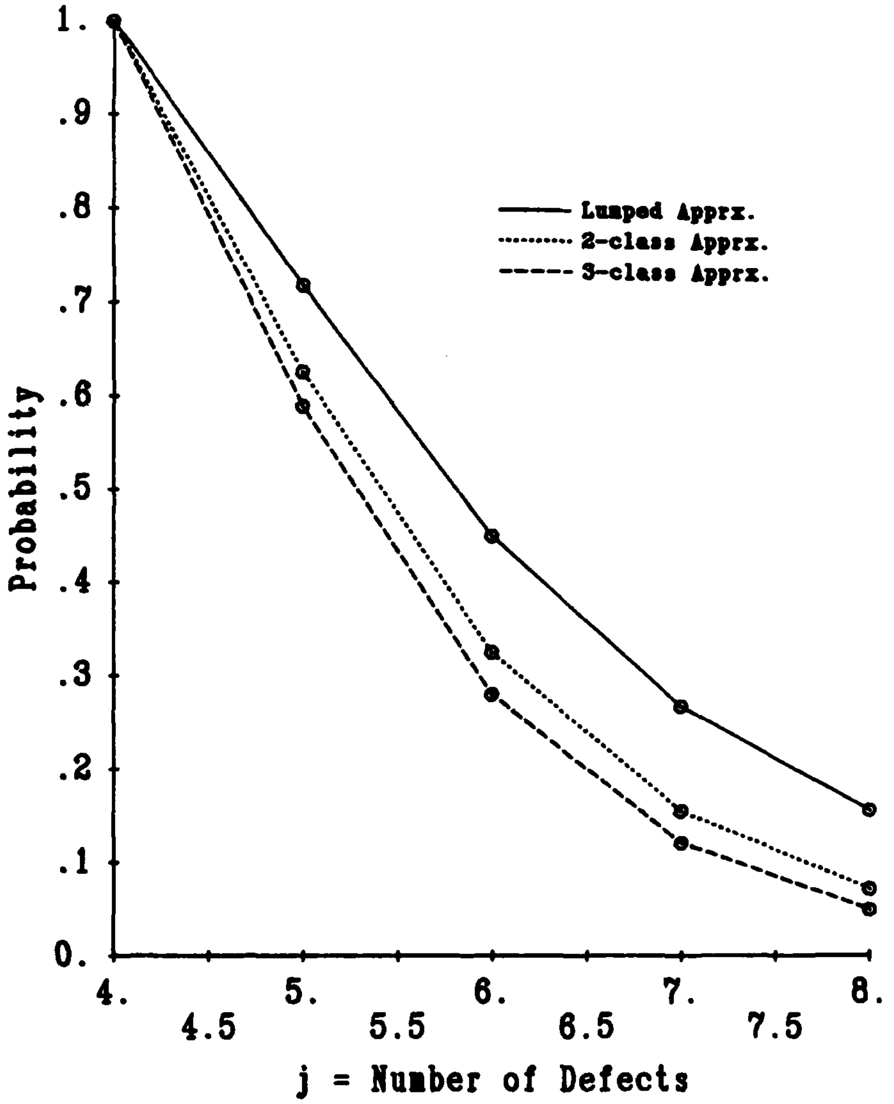


Figure A1.1 - Probability of j Defects Clustering in 4 of 16 PEs

VITA

VITA

Kye Sherrick Hedlund was born in Yonkers, New York on December 2, 1953. He graduated *cum laude* with Distinction in Mathematics from Boston University with a Bachelor of Arts in Mathematics in 1975. Mr. Hedlund has worked at IBM's Thomas J. Watson Research Center, Argonne National Laboratory, MIT Artificial Intelligence Laboratory and Call Data, Inc. Purdue University awarded him a Master of Science in Computer Science in 1979 and a Doctor of Philosophy in 1982. He is currently an Assistant Professor in the Computer Sciences Department of the University of North Carolina at Chapel Hill. While at Purdue he was employed as a teaching assistant and a research assistant. His high score at PACMAN is 196,400.